Missing Authorization Assistance – ZSU53

Applies to:

All SAP releases after ECC 5.0. For more information, visit the ABAP homepage.

Summary

This tool was developed to simplify the process of finding the appropriate profiles/roles for users to get the job done! Just execute this program and it will list all the profiles/roles which contain the missing (last failed) authorization! Assign any identified profile/role found to the user to get going with the work.

Author: Asim Rasheed Mian

Company: SAP Americas Inc.

Created on: 22 October 2009

Author Bio

Asim Rasheed Mian is a Business Application Analyst with SAP Americas (IT), working from Newtown Square, PA, USA since 2007 and has been doing ABAP development/consulting since 2003.

Table of Contents

Description

After coming across an authorization error, one commom practice is to run the transaction SU53, taking the screen shot of the same and sending it to e.g. a Basis Administrator, who in turn would analyze which exact authorization is missing (e.g. with transaction SUIM) to identify a role or profiles that contains the authorization so the user could carry on with the actual work.

The idea for this tool is to make this process easier for the users (or the Authorization Administrator and expedite the process of determining an appropriate profile for the users to get the job done! The program described below, looks similar to SU53, however it will list all the roles/profiles which contain the specific authorization that was missing! The tree display can also show the users assigned to these profiles/roles.

The program only helps in identifying a role/profile that solves a specific authorization issue. It does not touch on how to request these roles/profiles.

Pre-requisites

- Copy the original program behind SU53 (SAPMS01G) into customer name space. Include user interface and screens when you copy and activate all the objects, so you won't have to create the screen and GUI status manually.
- ZSU53 uses the same include as SU53, so you don't need to copy the include.
- Now copy the code provided below into the newly copied main program **Z**SAPMS01G, activate and execute!

Note: The code below was not tested with releases below ECC 5.0

How it works

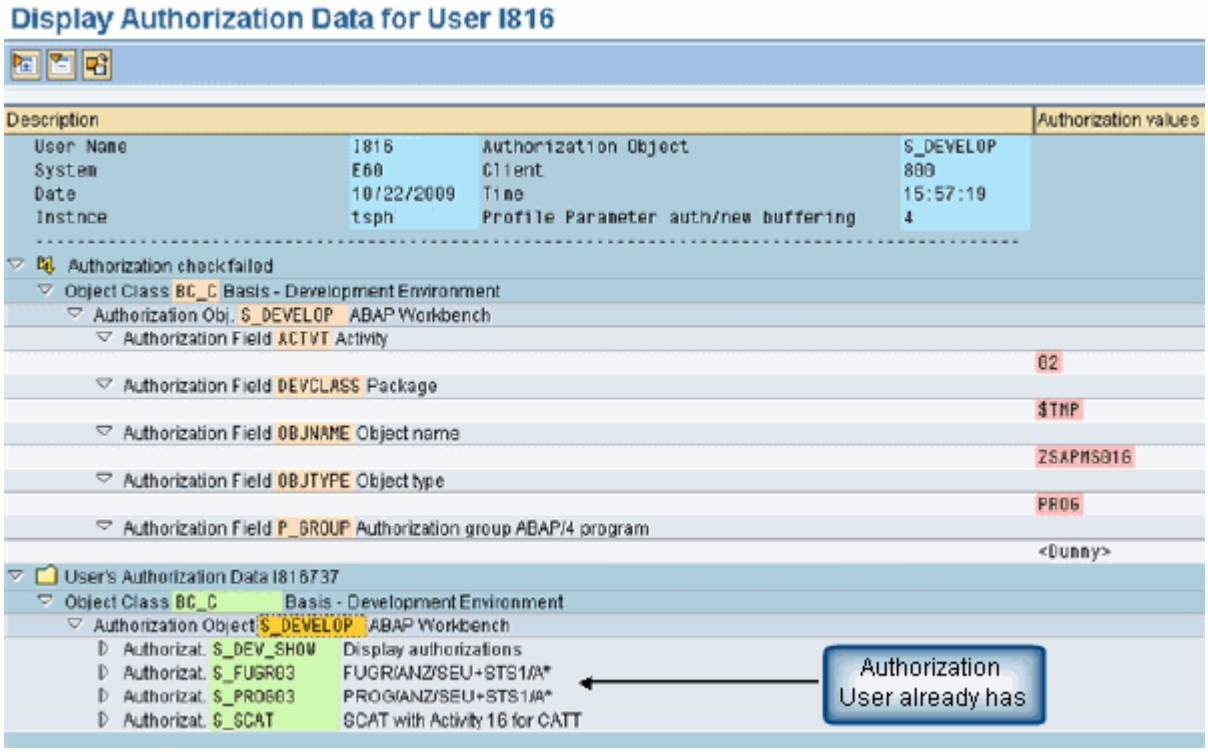
The basic principal of transaction SU53 remains unchanged the code reads the missing (last failed) authorization information from the parameters (XU1, XU2 and XU7) and saving them in table USR07. This entry is the key to all the look ups being done in the program, starting from table UST12 then filtering all the information per fields and values supplied in the USR07 structure, by keeping only those entries in the table UST12 which fit the required authorization criteria completely.

For example, if the user came across the authorization error for object S_DEVELOP and further attributes are activity '02', package 'ZARM', object type 'PROG' etc. Authorizations which contains value '02' but for another package or object type will be removed from the dataset, but the records where the activity is '02' and package is '*' or 'Z*' or that match the exact attributes will be kept.

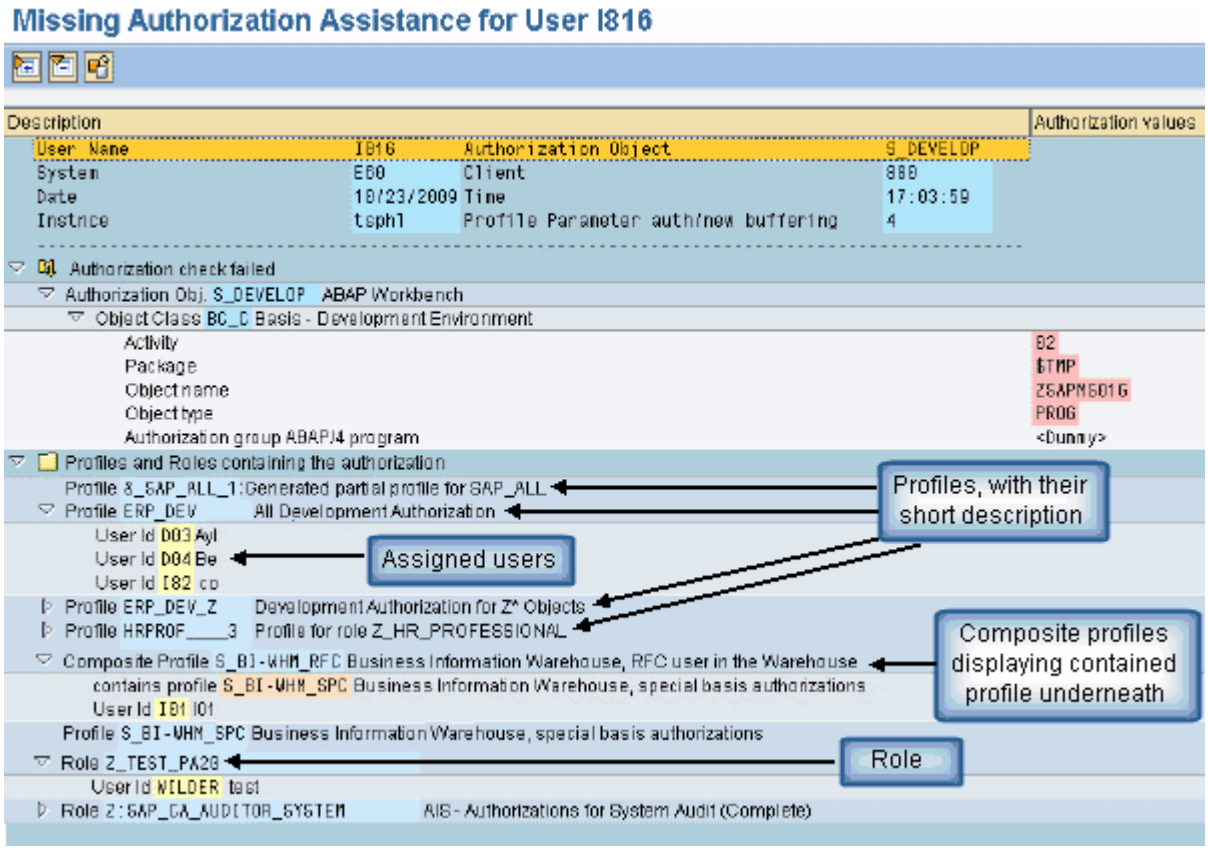
Once the final list of authorization values is determined the program looks up corresponding profiles containing these authorizations through table UST10S. At the same time any composite profiles from UST10C, which contains the single profiles looked up in UST10S are also pulled in. Finally the users assigned to these profiles/roles are also read from table UST04.

Here are two screen shots of the classic SU53 and ZSU53 (the missing authorization assistance program) for you to compare.

Transaction SU53 – Program SAPMS01G



New Transaction ZSU53 – Program ZSAPMS01G



These are the six major new forms introduced in the program:

- GET_RELATED_AUTH
- DIST_AUTH
- APPEND_AUTH
- COMP_AUTH
- AUTH2PROF
- FIND_PROFILES

The form “GET_RELATED_AUTH” being the main one which in turn calls the others and fills the internal table “it_all”, which is then used to fill the nodes and items of the tree for display.

In this specific implementation the variable(p_user was used to control whether the user would be able to see these assigned users or not. If initial, the user determination is skipped, if set to “X” (default currently) the users will be displayed.

Sample code

```
*&---------------------------------------------------------------------*
*& Report  ZSAPMS01G
*&---------------------------------------------------------------------*
*ZSU53 (ZSAPMS01G) - Is basic replica of SU53 along with added valuable
*      list of Profiles which actually has the required authorization.
* Created by: Asim Rasheed Mian
* Concept by: Martin Lang
* Create Date: May-28-2008
*&---------------------------------------------------------------------*

PROGRAM sapms01g MESSAGE-ID 01 NO STANDARD PAGE HEADING
LINE-SIZE 130.

INCLUDE ms01ctco.

TABLES: usr07, tobjt, dfies.
TABLES: tobj, tobct.

TABLES: agr_prof.
DATA lv_url(255) TYPE c.
DATA lv_langu(2) TYPE c.
DATA lv_sysid   TYPE sysysid.
DATA auth(30) TYPE c.
DATA profile_found(1) TYPE c.


DATA:   g_buffer_method(10).

DATA:   g_bname       LIKE usr07-bname, "Current user (global variable)
        g_objct       LIKE usr07-objct, "Current auth. object
        g_refuser     LIKE usrefus-refuser,
        g0010_bname   LIKE usr07-bname, "Screen fields
        g0010_objct   LIKE usr07-objct,
        g0010_entries,
        g0010_text(40),
        g_ok_code     LIKE sy-ucomm,
        g_datum       LIKE sy-datum,
        g_zeit        LIKE sy-uzeit.

DATA: excl TYPE TABLE OF sy-ucomm.

DATA: g_result_item_key_table  TYPE treemiks,
      g_find_string            TYPE string,
      wa_url(132).  " TYPE zsupport_http_e.

DATA:   BEGIN OF usr07key,
          objct LIKE usr07-objct,
          fiel0 LIKE usr07-fiel0,
          fiel1 LIKE usr07-fiel0,
          fiel2 LIKE usr07-fiel0,
          fiel3 LIKE usr07-fiel0,
          fiel4 LIKE usr07-fiel0,
          fiel5 LIKE usr07-fiel0,
          fiel6 LIKE usr07-fiel0,
          fiel7 LIKE usr07-fiel0,
          fiel8 LIKE usr07-fiel0,
```

```
          fiel9 LIKE usr07-fiel0,
        END OF usr07key.

DATA:   BEGIN OF usr07val1,
          val01 LIKE usr07-val01,
          val02 LIKE usr07-val01,
          val03 LIKE usr07-val01,
          val04 LIKE usr07-val01,
          val05 LIKE usr07-val01,
          val06 LIKE usr07-val01,
        END OF usr07val1.

DATA:   BEGIN OF usr07val2,
          val07 LIKE usr07-val01,
          val08 LIKE usr07-val01,
          val09 LIKE usr07-val01,
          val10 LIKE usr07-val01,
        END OF usr07val2.

DATA: it_ust12 LIKE TABLE OF ust12 WITH HEADER LINE,
      BEGIN OF wa_auth,
        auth  LIKE ust12-auth,
        bis   LIKE ust12-bis,
        von   LIKE ust12-von,
      END OF wa_auth,
      it_auth1 LIKE TABLE OF wa_auth WITH HEADER LINE,
      it_auth2 LIKE TABLE OF wa_auth WITH HEADER LINE,
      it_auth3 LIKE TABLE OF wa_auth WITH HEADER LINE,
      it_auth4 LIKE TABLE OF wa_auth WITH HEADER LINE,
      it_auth5 LIKE TABLE OF wa_auth WITH HEADER LINE,
      it_auth6 LIKE TABLE OF wa_auth WITH HEADER LINE,
      it_auth7 LIKE TABLE OF wa_auth WITH HEADER LINE,
      it_auth8 LIKE TABLE OF wa_auth WITH HEADER LINE,
      it_auth9 LIKE TABLE OF wa_auth WITH HEADER LINE,
      it_auth10 LIKE TABLE OF wa_auth WITH HEADER LINE,
      level(2) TYPE n.

DATA: BEGIN OF lt_ust04 OCCURS 0,
        bname TYPE bname,
        profile TYPE xuprofile,
        vorna(40),
        nachn(40),
*        bukrs TYPE bukrs,
      END OF lt_ust04,
      ls_ust04 LIKE LINE OF lt_ust04,
      lt_ust10c TYPE TABLE OF ust10c,
      ls_ust10c LIKE LINE OF lt_ust10c,
      lv_before TYPE i,
      lv_after TYPE i,
      BEGIN OF it_profn OCCURS 0,
        profn LIKE ust10s-profn,
        auth LIKE ust12-auth,
        subprof LIKE ust10c-subprof,
      END OF it_profn,

      it_auth LIKE TABLE OF wa_auth WITH HEADER LINE,
      BEGIN OF wa_prof,
        profile LIKE ust04-profile,
        bname LIKE ust04-bname,
```

```
        name(40) TYPE c,                                    "user name
        parprof LIKE ust04-profile,
        costc LIKE sadrp-costc,
        abtei LIKE sadrp-abtei,
        level(2) TYPE n,
      END OF wa_prof, "main table with all data

      it_profiles LIKE TABLE OF wa_prof WITH HEADER LINE,
      it_all LIKE TABLE OF wa_prof WITH HEADER LINE,
      it_users LIKE TABLE OF wa_prof,

      p_user VALUE 'X',  "variable to display user
      lv_s_date LIKE sy-datum,
      lv_s_time LIKE sy-uzeit,
      lv_s_stamp LIKE tzonref-tstamps,
      lv_e_date LIKE sy-datum,
      lv_e_time LIKE sy-uzeit,
      lv_e_stamp LIKE tzonref-tstamps,

      lv_diff_time LIKE sy-uzeit.

DATA: fcode TYPE TABLE OF sy-ucomm.

DATA: BEGIN OF ls_field_value,
        field LIKE tobj-fiel1,
        value LIKE usr07-val01,
      END OF ls_field_value.

START-OF-SELECTION.

* Get authorization buffer method
  CALL 'C_SAPGPARAM'
       ID 'NAME' FIELD 'auth/new_buffering'
       ID 'VALUE' FIELD g_buffer_method.

* Get result of last authorization check
  GET PARAMETER ID 'XU1' FIELD usr07key.                    "#EC EXISTS
  GET PARAMETER ID 'XU2' FIELD usr07val1.                   "#EC EXISTS
  GET PARAMETER ID 'XU7' FIELD usr07val2.                   "#EC EXISTS
* Store result of last authorization check
  CLEAR usr07.
  usr07-bname = sy-uname.
  MOVE-CORRESPONDING usr07key  TO usr07.
  MOVE-CORRESPONDING usr07val1 TO usr07.
  MOVE-CORRESPONDING usr07val2 TO usr07.
  MODIFY usr07.

* Tree control view of SU53
  g_bname = sy-uname.
  g_objct = usr07key-objct.
  PERFORM display_result_tree.

*---------------------------------------------------------------------*
*       FORM display_other_user                                       *
*---------------------------------------------------------------------*
FORM display_other_user.

  DATA rc LIKE sy-subrc.
```

```
  g0010_bname = g_bname.
  g0010_objct = g_objct.
  CALL SCREEN 10
    STARTING AT 10 10.
  IF g_ok_code = 'OK'.
    PERFORM display_result_tree.
  ENDIF.
ENDFORM.                    "display_other_user

*&---------------------------------------------------------------------*
*&      Form  auth_check_display_user
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->VALUE      text
*----------------------------------------------------------------------*
FORM auth_check_display_user USING value
                                   msgty LIKE sy-msgty
                                   rc    LIKE sy-subrc.
*   Authorization check to display the user data
  DATA: bname      LIKE usr02-bname,
        user_class LIKE usr02-class,
        error      LIKE sy-subrc..

  bname = value.
  CLEAR rc.

  IF sy-uname NE bname.                                     "#EC *
    CLEAR user_class.
    SELECT SINGLE class FROM usr02 INTO user_class
      WHERE bname = bname.
    IF sy-subrc NE 0.
      error = 4.
    ENDIF.
    PERFORM auth_check(saplsuu0)
          USING obj_group user_class space act_show rc.
    IF rc <> 0.
*     Keine Berechtigung zum Anzeigen
      MESSAGE  ID '01' TYPE msgty NUMBER '512' WITH user_class.
      rc = 4.
      EXIT.
    ENDIF.

    IF error = 4.
*     Benutzer existiert nicht
      MESSAGE ID '01' TYPE msgty NUMBER '124' WITH bname.
      rc = 4.
      EXIT.
    ENDIF.
  ENDIF.
ENDFORM.                    "auth_check_display_user

*&---------------------------------------------------------------------*
*&      Module  STATUS_0010  OUTPUT
*&---------------------------------------------------------------------*
MODULE status_0010 OUTPUT.
  SET PF-STATUS 'POPUP'.
  SET TITLEBAR 'SEL'.
```

```
ENDMODULE.                 " STATUS_0010  OUTPUT

*&---------------------------------------------------------------------*
*&      Module  check_bname  INPUT
*&---------------------------------------------------------------------*
MODULE check_bname INPUT.
  IF sy-ucomm NE 'CANC'.
    PERFORM check_bname.
  ENDIF.
ENDMODULE.                    "check_bname INPUT
*
FORM check_bname.
  DATA: rc    LIKE sy-subrc.
  PERFORM auth_check_display_user USING g0010_bname 'E' rc.
  CHECK rc = 0.

  CLEAR: usr07, usr07key, usr07val1, usr07val2.
  SELECT SINGLE * FROM usr07
    WHERE bname = g0010_bname.
  MOVE-CORRESPONDING usr07 TO usr07key.
  MOVE-CORRESPONDING usr07 TO usr07val1.
  MOVE-CORRESPONDING usr07 TO usr07val2.
  g0010_objct = usr07key-objct.
ENDFORM.                   " check_bname  INPUT

*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_0010  INPUT
*&---------------------------------------------------------------------*
MODULE user_command_0010 INPUT.
  CASE g_ok_code.
    WHEN 'CHECK'.
    WHEN 'OK'.
      IF g0010_bname <> '*'.
        g_bname = g0010_bname.
        g_objct = g0010_objct.
        LEAVE TO SCREEN 0.
      ELSE.
        MESSAGE s124(01) WITH g0010_bname." Benutzer existiert nicht
      ENDIF.
    WHEN 'CANC'.
      LEAVE TO SCREEN 0.
  ENDCASE.
ENDMODULE.                 " USER_COMMAND_0010  INPUT


*&---------------------------------------------------------------------*
*&      Form  values_read
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*

FORM values_read.                                           "#EC CALLED

  DATA: repid LIKE sy-repid,
        dynnr LIKE sy-dynnr,
        dynpfields TYPE TABLE OF dynpread WITH HEADER LINE.

  repid = sy-repid.
  dynnr = sy-dynnr.
```

```
  CLEAR: dynpfields, dynpfields[].
  dynpfields-fieldname  = 'G0010_BNAME'.
  APPEND dynpfields.

  CALL FUNCTION 'DYNP_VALUES_READ'
    EXPORTING
      dyname               = repid
      dynumb               = dynnr
    TABLES
      dynpfields           = dynpfields
    EXCEPTIONS
      invalid_abapworkarea = 1
      invalid_dynprofield  = 2
      invalid_dynproname   = 3
      invalid_dynpronummer = 4
      invalid_request      = 5
      no_fielddescription  = 6
      undefind_error       = 7
      OTHERS               = 8.
  IF sy-subrc <> 0.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*         WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
  ENDIF.
  READ TABLE dynpfields INDEX 1.
  IF sy-subrc = 0.
    g0010_bname = dynpfields-fieldvalue.
    SET LOCALE LANGUAGE space.
    TRANSLATE g0010_bname TO UPPER CASE.
  ENDIF.

ENDFORM.                    " VALUES_READ

*&---------------------------------------------------------------------*
*&      Module  usr07_bname_value_help  INPUT
*&---------------------------------------------------------------------*
MODULE usr07_bname_value_help INPUT.
  PERFORM usr07_bname_value_help.
ENDMODULE.                    "usr07_bname_value_help INPUT
*
FORM usr07_bname_value_help.

  DATA: rc    LIKE sy-subrc,
        repid LIKE sy-repid,
        dynnr LIKE sy-dynnr,
        dynpfields TYPE TABLE OF dynpread WITH HEADER LINE.

  DATA: window_title(80),
        BEGIN OF value_tab OCCURS 0,
          bname LIKE usr07-bname,
          objct LIKE usr07-objct,
        END OF value_tab,
        return_tab TYPE TABLE OF ddshretval WITH HEADER LINE,
        value_ LIKE help_info-fldvalue.

  RANGES value FOR usr07-bname.

  repid = sy-repid.
  dynnr = sy-dynnr.
  CLEAR: dynpfields, dynpfields[].
```

```
  dynpfields-fieldname  = 'G0010_BNAME'.
  APPEND dynpfields.

  CALL FUNCTION 'DYNP_VALUES_READ'
    EXPORTING
      dyname               = repid
      dynumb               = dynnr
    TABLES
      dynpfields           = dynpfields
    EXCEPTIONS
      invalid_abapworkarea = 1
      invalid_dynprofield  = 2
      invalid_dynproname   = 3
      invalid_dynpronummer = 4
      invalid_request      = 5
      no_fielddescription  = 6
      undefind_error       = 7
      OTHERS               = 8.

  IF sy-subrc <> 0.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*         WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
  ENDIF.

  READ TABLE dynpfields INDEX 1.

  IF sy-subrc = 0.
    IF dynpfields-fieldvalue IS INITIAL.
      value-sign   = 'I'.
      value-option = 'CP'.
      value-low = '*'.
      APPEND value.
    ELSE.
      g0010_bname = dynpfields-fieldvalue.
      SET LOCALE LANGUAGE space.
      TRANSLATE g0010_bname TO UPPER CASE.
      value-sign   = 'I'.
      value-option = 'CP'.
      value-low    = g0010_bname.
      APPEND value.
    ENDIF.
  ENDIF.


  SELECT bname objct FROM usr07 INTO TABLE value_tab
     WHERE bname IN value
       AND objct NE space.

  IF NOT value_tab[] IS INITIAL.
    value_ = g0010_bname.

    window_title = 'Users with Failed Authorization Checks'(023). "#EC *
    CALL FUNCTION 'F4IF_INT_TABLE_VALUE_REQUEST'
      EXPORTING
        window_title          = window_title
*     DDIC_STRUCTURE        = 'USR07'
        retfield              = 'BNAME'
*     PVALKEY               = ' '
*     DYNPPROG              = ' '
```

```
*      DYNPNR                  = ' '
*      DYNPROFIELD             = ' '
*      STEPL                   = 0
*      WINDOW_TITLE            =
         value                   = value_
         value_org               = 'S'
*      MULTIPLE_CHOICE         = ' '
*      DISPLAY                 = ' '
*      CALLBACK_PROGRAM        = ' '
*      CALLBACK_FORM           = ' '
       TABLES
         value_tab               = value_tab
*      FIELD_TAB               =
         return_tab              = return_tab
*      DYNPFLD_MAPPING         =
       EXCEPTIONS
         parameter_error         = 1
         no_values_found         = 2
         OTHERS                  = 3.

    IF sy-subrc <> 0.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*         WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
    ENDIF.

    READ TABLE return_tab INDEX 1.

    IF sy-subrc EQ 0.
      g0010_bname = return_tab-fieldval.
      READ TABLE value_tab WITH KEY bname = g0010_bname.
      g0010_objct = value_tab-objct.

      repid = sy-repid.
      dynnr = sy-dynnr.
      CLEAR: dynpfields, dynpfields[].
      dynpfields-fieldname  = 'G0010_OBJCT'.
      dynpfields-fieldvalue = g0010_objct.
      APPEND dynpfields.

      CALL FUNCTION 'DYNP_VALUES_UPDATE'
        EXPORTING
          dyname               = repid
          dynumb               = dynnr
        TABLES
          dynpfields           = dynpfields
        EXCEPTIONS
          invalid_abapworkarea = 1
          invalid_dynprofield  = 2
          invalid_dynproname   = 3
          invalid_dynpronummer = 4
          invalid_request      = 5
          no_fielddescription  = 6
          undefind_error       = 7
          OTHERS               = 8.

      IF sy-subrc <> 0.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*         WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
      ENDIF.
```

```
    ENDIF.
  ELSE.
    MESSAGE  i802(dh).     " Keine Werte zu dieser Selektion
    EXIT.
  ENDIF.
ENDFORM.                  " usr07_bname_value_help  INPUT


*---------------------------------------------------------------------*
*  Tree control
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
*     Global DATA
*---------------------------------------------------------------------*

CLASS lcl_application DEFINITION DEFERRED.
CLASS cl_gui_cfw DEFINITION LOAD.

* No own DDIC structure up to now - Reuse of MTREEITM
TYPES: item_type       TYPE mtreeitm.

TYPES: item_table_type TYPE STANDARD TABLE OF item_type
         WITH DEFAULT KEY.

DATA: g_application      TYPE REF TO lcl_application,
      g_custom_container TYPE REF TO cl_gui_custom_container,
      g_tree             TYPE REF TO cl_list_tree_model.

* Store keys of nodes (which can be expanded)
TYPES: BEGIN OF gt_nodes_struct,
         node_key TYPE tm_nodekey,
         isfolder(1) TYPE c,
         expander(1) TYPE c,
         objct    TYPE ust12-objct,
         auth     TYPE ust12-auth,
         profile  TYPE ust04-profile,
         agr_name TYPE agr_1016-agr_name,
         refuser  TYPE usrefus-refuser,
         field    TYPE dfies-fieldname,
       END OF gt_nodes_struct.
DATA: gt_nodes TYPE SORTED TABLE OF gt_nodes_struct
      WITH UNIQUE KEY node_key.

* Unique numeric key for nodes
DATA: g_node_key_num   TYPE i,
       g_node_key      TYPE i,
*       g_node_key_auth TYPE i,
*       g_node_key_auth1 TYPE i,
*       g_node_key_auth2 TYPE i,
*       g_node_key_prof TYPE i,
*       g_node_key_ref TYPE i,
       g_node_key_role TYPE i.


* Authorization date of the user
DATA: BEGIN OF gt_profiles OCCURS 0,
        profile  LIKE ust04-profile,
```

```
        refuserflag(1),
        typ      LIKE usr10-typ,
        ptext    LIKE usr11-ptext,
        agr_name LIKE agr_1016-agr_name,
        atext    LIKE agr_texts-text,
      END OF gt_profiles.

DATA: BEGIN OF gt_auths OCCURS 0,
        objct    LIKE ust10s-objct,
        auth     LIKE ust10s-auth,
        profile  LIKE ust04-profile, "Source of authorization
        userbuffer(1) TYPE c, "Authorization in user buffer only
      END OF gt_auths.

DATA: BEGIN OF gt_ust12 OCCURS 0,
        objct LIKE ust12-objct,
        auth  LIKE ust12-auth,
        field LIKE ust12-field,
        von   LIKE ust12-von,
        bis   LIKE ust12-bis,
      END OF gt_ust12.

DATA: usebgcolor(1) VALUE ' '. " X = light grey background


*&---------------------------------------------------------------------*
*&      Form  display_result_tree
*&---------------------------------------------------------------------*
*       -->UNAME      text
*----------------------------------------------------------------------*
FORM display_result_tree.

  SET SCREEN 100.

ENDFORM.                    "display_result_tree

*----------------------------------------------------------------------*
*       CLASS lcl_application DEFINITION
*----------------------------------------------------------------------*
CLASS lcl_application DEFINITION.
  PUBLIC SECTION.
    METHODS:
      handle_node_double_click
        FOR EVENT node_double_click
        OF cl_list_tree_model
        IMPORTING node_key,
*      handle_expand_no_children
*        FOR EVENT expand_no_children
*        OF cl_list_tree_model
*        IMPORTING node_key,
      handle_item_double_click
        FOR EVENT item_double_click
        OF cl_list_tree_model
        IMPORTING node_key item_name,
      handle_button_click
        FOR EVENT button_click
        OF cl_list_tree_model
        IMPORTING node_key item_name,
      handle_link_click
```

```
        FOR EVENT link_click
        OF cl_list_tree_model
        IMPORTING node_key item_name,
      handle_checkbox_change
        FOR EVENT checkbox_change
        OF cl_list_tree_model
        IMPORTING node_key item_name checked.
ENDCLASS.                    "LCL_APPLICATION DEFINITION

*---------------------------------------------------------------------*
*       CLASS LCL_APPLICATION IMPLEMENTATION
*---------------------------------------------------------------------*
CLASS lcl_application IMPLEMENTATION.

  METHOD  handle_node_double_click.
    " USING node_key
    " this method handles the node double click event of the tree
    " control instance
  ENDMETHOD.                    "HANDLE_NODE_DOUBLE_CLICK

  METHOD  handle_item_double_click.
    " USING node_key item_name
    " this method handles the item double click event of the tree
    " control instance

    DATA: wa_node_table LIKE LINE OF gt_nodes.

    READ TABLE gt_nodes INTO wa_node_table WITH KEY node_key = node_key.
    CHECK sy-subrc = 0.

*   Navigation to profile (list instead of control)
    IF     NOT wa_node_table-profile  IS INITIAL.
      AUTHORITY-CHECK OBJECT 'S_USER_PRO'
                ID 'PROFILE' FIELD wa_node_table-profile
                ID 'ACTVT'   FIELD '03'.
      IF sy-subrc = 0.
        CALL FUNCTION 'SUSR_PROF_DISPLAY_WITH_AUTHS'
          EXPORTING
            profile               = wa_node_table-profile
*           P_STATE               = 'A'
          EXCEPTIONS
            prof_dont_exist       = 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.
      ENDIF.

    ENDIF.
  ENDMETHOD.                    "HANDLE_ITEM_DOUBLE_CLICK

  METHOD  handle_link_click.
    " USING node_key item_name
    " this method handles the link click event of the tree
    " control instance
  ENDMETHOD.                    "HANDLE_LINK_CLICK
```

```
  METHOD  handle_button_click.
    " USING node_key item_name
    " this method handles the button click event of the tree
    " control instance
  ENDMETHOD.                    "HANDLE_BUTTON_CLICK

  METHOD  handle_checkbox_change.
    " USING node_key item_name
    " this method handles the checkbox_change event of the tree
    " control instance
  ENDMETHOD.                    "HANDLE_CHECKBOX_CHANGE


ENDCLASS.                    "LCL_APPLICATION IMPLEMENTATION

*&---------------------------------------------------------------------*
*&      Module  PBO_0100  OUTPUT
*&---------------------------------------------------------------------*
MODULE pbo_0100 OUTPUT.

  SET PF-STATUS 'TREE' EXCLUDING fcode.
  SET TITLEBAR 'USR' WITH g_bname.

  g_datum = sy-datum.
  g_zeit  = sy-uzeit.
  IF g_tree IS INITIAL.
    " The Tree Control has not been created yet.
    " Create a Tree Control and insert nodes into it.
    PERFORM create_and_init_tree.
  ENDIF.
ENDMODULE.                 " PBO_0100  OUTPUT

*&---------------------------------------------------------------------*
*&      Form  CREATE_AND_INIT_TREE
*&---------------------------------------------------------------------*
* The Tree Control has not been created yet.
* Create a Tree Control and insert nodes into it.
*----------------------------------------------------------------------*
FORM create_and_init_tree.

  DATA: node_table      TYPE treemlnota,
        item_table      TYPE treemlitac,
        nodes_to_expand TYPE treemnotab,
        events          TYPE cntl_simple_events,
        event           TYPE cntl_simple_event,
        lv_top_node     LIKE LINE OF nodes_to_expand.

  CHECK g_tree IS INITIAL.

  CALL FUNCTION 'SAPGUI_PROGRESS_INDICATOR'
    EXPORTING
*     PERCENTAGE       = 0
      text             = 'Initialization'(026)              "#EC *
           .

* create the application object
* this object is needed to handle the ABAP Objects Events of
* Controls
  CREATE OBJECT g_application.
```

```
* create a container for the tree control
  CREATE OBJECT g_custom_container
   EXPORTING     " the container is linked to the custom control with the
            " name 'TREE_CONTAINER' on the dynpro

    container_name = 'TREE_CONTAINER'
    EXCEPTIONS
    cntl_error = 1
    cntl_system_error = 2
    create_error = 3
    lifetime_error = 4
    lifetime_dynpro_dynpro_link = 5.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.

  DATA:l_hierarchy_header TYPE treemhhdr,
       l_list_header      TYPE treemlhdr.

  l_hierarchy_header-heading   = 'Description'(017).        "#EC *
  l_hierarchy_header-tooltip   = ' '.

  l_list_header-heading        = 'Authorization Values'(072). "#EC *
  l_list_header-tooltip        = ' '.

* create a list tree control
  CREATE OBJECT g_tree
    EXPORTING
      node_selection_mode         = cl_list_tree_model=>node_sel_mode_single
      item_selection              = 'X'
      with_headers                = 'X'
      hierarchy_header            = l_hierarchy_header
      list_header                 = l_list_header
    EXCEPTIONS
      illegal_node_selection_mode = 1.
  IF sy-subrc <> 0.
    MESSAGE a001.
  ENDIF.

* define the events which will be passed to the backend
  " node double click
  event-eventid = cl_list_tree_model=>eventid_node_double_click.
  event-appl_event = 'X'.                                    "
  APPEND event TO events.

  " item double click
  event-eventid = cl_list_tree_model=>eventid_item_double_click.
  event-appl_event = 'X'.
  APPEND event TO events.


  " link click
  event-eventid = cl_list_tree_model=>eventid_link_click.
  event-appl_event = 'X'.
  APPEND event TO events.

  " button click
```

```
  event-eventid = cl_list_tree_model=>eventid_button_click.
  event-appl_event = 'X'.
  APPEND event TO events.

  " checkbox change
  event-eventid = cl_list_tree_model=>eventid_checkbox_change.
  event-appl_event = 'X'.
  APPEND event TO events.

  CALL METHOD g_tree->set_registered_events
    EXPORTING
      events                    = events
    EXCEPTIONS
      illegal_event_combination = 1
      unknown_event             = 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.

* assign event handlers in the application class to each desired event
  SET HANDLER g_application->handle_node_double_click  FOR g_tree.
  SET HANDLER g_application->handle_item_double_click  FOR g_tree.
*  SET HANDLER g_application->handle_expand_no_children FOR g_tree.
  SET HANDLER g_application->handle_link_click         FOR g_tree.
  SET HANDLER g_application->handle_button_click       FOR g_tree.
  SET HANDLER g_application->handle_checkbox_change    FOR g_tree.

* Add nodes to the tree control
* NOTE: the tree control does not store data at the backend. If an
* application wants to access tree data later, it must store the
* tree data itself.

  PERFORM build_node_and_item_table USING node_table item_table
                                          nodes_to_expand
                                          lv_top_node.

  CALL METHOD g_tree->add_nodes
    EXPORTING
      node_table          = node_table
    EXCEPTIONS
      error_in_node_table = 1
      OTHERS              = 2.

  CALL METHOD g_tree->add_items
    EXPORTING
      item_table          = item_table
    EXCEPTIONS
      error_in_item_table = 1
      OTHERS              = 2.


  CALL METHOD g_tree->create_tree_control
    EXPORTING
      parent                      = g_custom_container
    EXCEPTIONS
      lifetime_error              = 1
      cntl_system_error           = 2
      create_error                = 3
```

```
      failed                      = 4
      tree_control_already_created = 5
      OTHERS                      = 6.

* Expand nodes describing errors
  IF NOT nodes_to_expand[] IS INITIAL.
    CALL METHOD g_tree->expand_nodes
      EXPORTING
        node_key_table = nodes_to_expand
      EXCEPTIONS
        OTHERS         = 1.
    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
              WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.

**   Scoll to top (expand_nodes scrolls to last expanded node)
    CALL METHOD g_tree->scroll
      EXPORTING
        scroll_command         = cl_list_tree_model=>scroll_home
      EXCEPTIONS
        failed                 = 1
        illegal_scroll_command = 2
        cntl_system_error      = 3
        OTHERS                 = 4.
    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
              WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.

  ENDIF.

*  if lv_top_node is NOT INITIAL.
*    CALL METHOD g_tree->SET_SELECTED_NODE
*      EXPORTING
*        node_key             = lv_top_node
**      EXCEPTIONS
**        control_not_existing = 1
**        control_dead         = 2
**        cntl_system_error    = 3
**        failed               = 4
**        node_not_found       = 5
**        others               = 6
*            .
*    IF sy-subrc <> 0.
**     MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
**                WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
*    ENDIF.
*  ENDIF.

  CALL METHOD g_tree->hierarchy_header_adjust_width
    EXCEPTIONS
      cntl_system_error = 1
      failed            = 2
      OTHERS            = 3.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.
```

```
ENDFORM.                    "CREATE_AND_INIT_TREE

*&---------------------------------------------------------------------*
*&      Form  build_node_and_item_table
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
FORM build_node_and_item_table
  USING
    node_table      TYPE treemlnota
    item_table      TYPE treemlitac
    nodes_to_expand TYPE treemnotab
    lv_top_node     LIKE LINE OF nodes_to_expand.

  DATA: node TYPE treemlnodt,
        item TYPE treemliten.

  DATA: wa_node_table LIKE LINE OF gt_nodes.

  DATA: field  LIKE usr07-fiel1,
        number TYPE i VALUE 0,
        rc     LIKE sy-subrc.

  DATA: BEGIN OF field_value OCCURS 10,
          field LIKE tobj-fiel1,
          value LIKE usr07-val01,
        END OF field_value.

* Variables to resolve multiple checked field in one statement
  DATA: count  TYPE i,
        count_char(2) TYPE c,
        p_field TYPE usr07-fiel1.

* Build the node table.

  CLEAR g_node_key_num.

* Caution: The nodes are inserted into the tree according to the order
* in which they occur in the table. In consequence, a node must not
* must not occur in the node table before its parent node.

* ---------------------------------------------------------------------
* Folder with key 'Header': Tree of previous authorization check
* ---------------------------------------------------------------------
  PERFORM header_data   USING node_table
                              item_table.

* ---------------------------------------------------------------------
* Folder with key 'Root1': Tree of previous authorization check
* ---------------------------------------------------------------------

*  CLEAR node.
  node-node_key = 'Root1'.                                  "#EC NOTEXT
  node-relatkey  = ' '.  " Special case: A root node has no parent node
  node-relatship = cl_list_tree_model=>relat_next_sibling.
```

```
  node-hidden    = ' '.          " The node is visible,
  node-disabled  = ' '.          " selectable,
  node-isfolder  = 'X'.          " a folder,
  node-n_image   = 'BNONE'.
  node-expander  = space.        " and a subtree.
  node-exp_image = 'BNONE'.      " Folder-/ Leaf-Symbol in state "open"
  APPEND node TO node_table.

  CLEAR wa_node_table.
  wa_node_table-node_key       = node-node_key.
  wa_node_table-isfolder       = node-isfolder.
  wa_node_table-expander       = node-expander.
  INSERT wa_node_table INTO TABLE gt_nodes.

  APPEND node-node_key TO nodes_to_expand.

  IF usr07key-objct IS INITIAL.

*   Sucessful authorization check

    CLEAR item.
    item-node_key = node-node_key.
    item-item_name = '01'.
    item-class = cl_list_tree_model=>item_class_text.
    CALL FUNCTION 'ICON_CREATE'
      EXPORTING
        name                  = 'ICON_CHECKED'
        text                  = ' '
        info                  = ' '
        add_stdinf            = ' '
      IMPORTING
        RESULT                = item-t_image
      EXCEPTIONS
        icon_not_found        = 1
        outputfield_too_short = 2
        OTHERS                = 3.
    IF sy-subrc <> 0.
      item-t_image = '@01@'.               " okay
    ENDIF.
    APPEND item TO item_table.

    CLEAR item.
    item-node_key   = node-node_key.
    item-item_name  = '02'.
    item-class      = cl_list_tree_model=>item_class_text.
    item-alignment  = cl_list_tree_model=>align_auto.
    item-font       = cl_list_tree_model=>item_font_prop.
    item-text       =
      'The last authorization check was successful'(076).    "#EC *
    APPEND item TO item_table.

  ELSE.

*   Not sucessful authorization check
    CLEAR item.
    item-node_key = node-node_key.
    item-item_name = '01'.
    item-class = cl_list_tree_model=>item_class_text.
*   item-length = 4. " the width of the item is 4 characters
```

```
*   item-ignoreimag = 'X'. " see documentation of Structure TREEV_ITEM
    CALL FUNCTION 'ICON_CREATE'
      EXPORTING
        name                  = 'ICON_FAILURE'
        text                  = ' '
        info                  = ' '
        add_stdinf            = ' '
      IMPORTING
        RESULT                = item-t_image
      EXCEPTIONS
        icon_not_found        = 1
        outputfield_too_short = 2
        OTHERS                = 3.
    IF sy-subrc <> 0.
      item-t_image = '@03@'.                " error
    ENDIF.
    item-text = 'Authorization check failed'(074).          "#EC *
    item-txtisqinfo = 'X'.
    APPEND item TO item_table.

    CLEAR item.
    item-node_key = node-node_key.
    item-item_name = '02'.
    item-class = cl_list_tree_model=>item_class_text. " Text Item
    " the with of the item is adjusted to its content (text)
    item-alignment = cl_list_tree_model=>align_auto.
    " use proportional font for the item
    item-font = cl_list_tree_model=>item_font_prop.
    item-text = 'Authorization check failed'(074).          "#EC *
    APPEND item TO item_table.

*   Folder with key 'Result': Result of previous authorization check

    CLEAR node.
    node-node_key   = 'Result'.                             "#EC NOTEXT
    node-relatkey   = 'Root1'.                              "#EC NOTEXT
    node-relatship  = cl_list_tree_model=>relat_last_child.
    node-n_image    = 'BNONE'.
    node-isfolder   = 'X'.
    APPEND node TO node_table.

    CLEAR wa_node_table.
    wa_node_table-node_key       = node-node_key.
    wa_node_table-isfolder       = node-isfolder.
    wa_node_table-expander       = node-expander.
    INSERT wa_node_table INTO TABLE gt_nodes.

    APPEND node-node_key TO nodes_to_expand.

    CLEAR item.
    item-node_key   = node-node_key.
    item-item_name  = '02'.
    item-class      = cl_list_tree_model=>item_class_text.
    item-alignment  = cl_list_tree_model=>align_auto.
    item-font       = cl_list_tree_model=>item_font_prop.
    item-text       = 'Authorization Object'(018).          "#EC *
    APPEND item TO item_table.

    CLEAR item.
```

```
    item-node_key   = node-node_key.
    item-item_name  = '03'.
    item-class      = cl_list_tree_model=>item_class_text.
*    item-alignment  = cl_list_tree_model=>align_auto.
    item-length     = 10.
    item-font       = cl_list_tree_model=>item_font_fixed.
    item-usebgcolor = usebgcolor. " light grey background
    item-
style      = 8. "cl_list_tree_model=>style_emphasized_a.    "COL_HEADING
    item-text       = usr07key-objct.
    APPEND item TO item_table.

    DATA l_item TYPE treemliten.
    DATA flag.

    CLEAR item.
    CLEAR flag.
    item-node_key = node-node_key.
    item-item_name = '04'.
    item-class = cl_list_tree_model=>item_class_text.
    item-alignment = cl_list_tree_model=>align_auto.
    item-font = cl_list_tree_model=>item_font_prop.
    SELECT SINGLE ttext FROM tobjt INTO l_item-text
           WHERE langu  = sy-langu
           AND   object = usr07key-objct.
    IF sy-subrc = 0.
      item-text = l_item-text.
    ELSE.
      item-usebgcolor = usebgcolor. " light grey background
      item-style      = 5.
      item-text = 'Authorization object does not exist'(044). "#EC *
      flag = 1.
    ENDIF.
    APPEND item TO item_table.

*   Folder with key 'Class': Result of previous authorization check
    IF flag <> 1.
      CLEAR node.
      node-node_key   = 'Class'.                              "#EC NOTEXT
      node-relatkey   = 'Result'.                             "#EC NOTEXT
      node-relatship  = cl_list_tree_model=>relat_last_child.
      node-n_image    = 'BNONE'.
      node-isfolder   = 'X'.
      APPEND node TO node_table.

      CLEAR wa_node_table.
      wa_node_table-node_key       = node-node_key.
      wa_node_table-isfolder       = node-isfolder.
      wa_node_table-expander       = node-expander.
      INSERT wa_node_table INTO TABLE gt_nodes.

      APPEND node-node_key TO nodes_to_expand.

      CLEAR item.
      item-node_key   = node-node_key.
      item-item_name  = '02'.
      item-class      = cl_list_tree_model=>item_class_text.
      item-alignment  = cl_list_tree_model=>align_auto.
      item-font       = cl_list_tree_model=>item_font_prop.
```

```
      item-text       = 'Object Class'(041).                "#EC *
      APPEND item TO item_table.

      DATA class TYPE string.
      CLEAR item.
      CLEAR class.
      item-node_key   = node-node_key.
      item-item_name  = '03'.
      item-class      = cl_list_tree_model=>item_class_text.
*     item-alignment  = cl_list_tree_model=>align_auto.
      item-length     = 4.
      item-font       = cl_list_tree_model=>item_font_fixed.
      item-usebgcolor = usebgcolor. " light grey background
      item-style      = 8. "cl_list_tree_model=>style_emphasized_a.
      "COL_HEADING
      SELECT SINGLE oclss FROM tobj INTO item-text
         WHERE   objct = usr07key-objct.
      class = item-text.
      APPEND item TO item_table.

      CLEAR item.
      item-node_key = node-node_key.
      item-item_name = '04'.
      item-class = cl_list_tree_model=>item_class_text.
      item-alignment = cl_list_tree_model=>align_auto.
      item-font = cl_list_tree_model=>item_font_prop.
      SELECT SINGLE ctext FROM tobct INTO item-text
         WHERE langu  = sy-langu
         AND   oclss = class.
      APPEND item TO item_table.


*   nodes with authorization field values

      CLEAR: number, field_value, field_value[], p_field, count.
      DO 10 TIMES VARYING field FROM usr07key-fiel0 NEXT usr07key-fiel1.
        IF field <> space.
          number = number + 1.
          field_value-field = field.
          CASE number.
            WHEN 1.
              field_value-value = usr07val1-val01.
            WHEN 2.
              field_value-value = usr07val1-val02.
            WHEN 3.
              field_value-value = usr07val1-val03.
            WHEN 4.
              field_value-value = usr07val1-val04.
            WHEN 5.
              field_value-value = usr07val1-val05.
            WHEN 6.
              field_value-value = usr07val1-val06.
            WHEN 7.
              field_value-value = usr07val2-val07.
            WHEN 8.
              field_value-value = usr07val2-val08.
            WHEN 9.
              field_value-value = usr07val2-val09.
            WHEN 10.
```

```
            field_value-value = usr07val2-val10.
        ENDCASE.
        APPEND field_value.
      ENDIF.
    ENDDO.
    SORT field_value BY field.
    LOOP AT field_value.
      IF p_field EQ field_value-field.
        count = count + 1.
      ELSE.
        CLEAR count.
      ENDIF.
      CALL FUNCTION 'AUTH_FIELD_GET_INFO'
        EXPORTING
          fieldname = field_value-field
        IMPORTING
          datel     = dfies-rollname
          lng       = dfies-outputlen
          rc        = rc
          text      = dfies-fieldtext
          inttype   = dfies-inttype.

*     Node with authorization field name and value

      CLEAR node.
      WRITE count TO count_char.
      CONCATENATE 'F_' field_value-field count_char       "#EC NOTEXT
        INTO node-node_key.
      node-relatkey   = 'Class'.                          "#EC NOTEXT
      node-relatship  = cl_list_tree_model=>relat_last_child.
      node-last_hitem = '1'.
      node-n_image    = 'BNONE'.
      node-isfolder   = ' '.
      APPEND node TO node_table.

      CLEAR wa_node_table.
      wa_node_table-node_key       = node-node_key.
      wa_node_table-isfolder       = node-isfolder.
      wa_node_table-expander       = node-expander.
      INSERT wa_node_table INTO TABLE gt_nodes.

      APPEND node-node_key TO nodes_to_expand.

      CLEAR item.
      item-node_key   = node-node_key.
      item-item_name  = '01'.
      item-class      = cl_list_tree_model=>item_class_text.
      item-alignment  = cl_list_tree_model=>align_auto.
      item-font       = cl_list_tree_model=>item_font_prop.
      item-text       = dfies-fieldtext.
      APPEND item TO item_table.

      CLEAR item.
      item-node_key   = node-node_key.
      item-item_name  = '02'.
      item-class      = cl_list_tree_model=>item_class_text.
      item-alignment  = cl_list_tree_model=>align_auto.
      item-font       = cl_list_tree_model=>item_font_fixed.
      item-usebgcolor = usebgcolor. " light grey background
```

```
        IF field_value-value <> space.
          item-style      = 5.
          item-text     = field_value-value.
        ELSE.
          item-text     = '<Dummy>'(020).                    "#EC *
        ENDIF.
        APPEND item TO item_table.
        p_field = field_value-field.
      ENDLOOP.

    ENDIF.
  ENDIF.

*ENDIF.
* ---------------------------------------------------------------------
* Folder with key 'Root2': Authorization data of the user
*  profiles containing the authorizations
* ---------------------------------------------------------------------
  DATA: wa_profiles   LIKE LINE OF gt_profiles,
        wa_auths      LIKE LINE OF gt_auths.

  DATA: auth_node_key  TYPE i,
        field_node_key TYPE i.

  DATA: l_wa_node_table LIKE LINE OF gt_nodes.

  IF NOT g_objct IS INITIAL.

    CALL FUNCTION 'SAPGUI_PROGRESS_INDICATOR'
      EXPORTING
*       PERCENTAGE       = 0
        text             = 'Reading authorization data...'(027) "#EC *
              .

*   Read authorization data of the user

    DATA: BEGIN OF authbuffer OCCURS 30,
            object    LIKE usr12-objct,
            auth      LIKE usr12-auth,
          END OF authbuffer.

*   Read authorization user buffer
    CLEAR authbuffer[].

*   Read profiles (master data)
    CLEAR: gt_profiles, gt_profiles[].

    PERFORM get_related_auth TABLES field_value.

    CLEAR: gt_auths, gt_auths[].

*   Folder with key 'Root2': Authorization data of the user

    CALL FUNCTION 'SAPGUI_PROGRESS_INDICATOR'
      EXPORTING
*       PERCENTAGE       = 0
        text             = 'Display...'(028)                  "#EC *
              .
```

```
    node-node_key  = usr07-
objct. "'Root2'.                                "#EC NOTEXT
    node-relatkey  = ' '.
    " Special case: A root node has no parent node
    node-relatship = cl_list_tree_model=>relat_next_sibling.
    node-hidden    = ' '.  " The node is visible,
    node-disabled  = ' '.  " selectable,
    node-isfolder  = 'X'.  " a folder.
    CLEAR node-n_image.   " Folder-/ Leaf-Symbol in state "closed":
    CLEAR node-exp_image. " Folder-/ Leaf-Symbol in state "open":
    CLEAR node-expander.                  " see below.
    APPEND node TO node_table.

    CLEAR wa_node_table.
    wa_node_table-node_key       = node-node_key.
    wa_node_table-isfolder       = node-isfolder.
    wa_node_table-expander       = node-expander.
    INSERT wa_node_table INTO TABLE gt_nodes.

    APPEND node-node_key TO nodes_to_expand.

    CLEAR item.
    item-node_key   = node-node_key.
    item-item_name  = '01'.
    item-class      = cl_list_tree_model=>item_class_text. " Text item
    " the with of the item is adjusted to its content (text)
    item-alignment  = cl_list_tree_model=>align_auto.
    " use proportional font for the item
    item-font       = cl_list_tree_model=>item_font_prop.
    item-text       = 'Profiles and Roles containing the authorization'. "#EC *
    APPEND item TO item_table.

    g_node_key_num = 20000.
    g_node_key_role = 500000.

*    CLEAR node.
*    node-node_key   = usr07-objct. "g_node_key_num. "#EC NOTEXT
*    node-relatkey   = 'Root2'.                          "#EC NOTEXT
*    node-relatship  = cl_list_tree_model=>relat_last_child.
*    node-hidden     = ' '.  " The node is visible,
*    node-disabled   = ' '.  " selectable,
*    node-isfolder   = 'X'.  " a folder.
*    node-n_image    = 'BNONE'.  "no icon
*    APPEND node TO node_table.
*
*    CLEAR wa_node_table.
*    wa_node_table-node_key       = node-node_key.
*    wa_node_table-isfolder       = node-isfolder.
*    wa_node_table-expander       = node-expander.
*    wa_node_table-objct          = usr07-objct."g_node_key_num. "
*    INSERT wa_node_table INTO TABLE gt_nodes.
*
*    APPEND node-node_key TO nodes_to_expand.
*
**   Items of authorization object
*    CLEAR item.
*    item-node_key   = node-node_key.
*    item-item_name  = '01'.
*    item-class      = cl_list_tree_model=>item_class_text.
```

```
*    item-alignment  = cl_list_tree_model=>align_auto.
*    item-font       = cl_list_tree_model=>item_font_prop.
*    item-text       = 'Authorization Object'(079).          "#EC *
*    APPEND item TO item_table.
*
*    CLEAR item.
*    item-node_key   = node-node_key.
*    item-item_name  = '02'.
*    item-class      = cl_list_tree_model=>item_class_text.
**   item-alignment  = cl_list_tree_model=>align_auto.
*    item-length     = 10.
*    item-font       = cl_list_tree_model=>item_font_fixed.
*    item-usebgcolor = usebgcolor. " light grey background
*    item-style      = 8. "cl_list_tree_model=>style_emphasized_a.
*    "COL_HEADING
*    item-text       = usr07-objct.
*    APPEND item TO item_table.
*
*    CLEAR item.
*    item-node_key  = node-node_key.
*    item-item_name = '03'.
*    item-class     = cl_list_tree_model=>item_class_text.
*    item-alignment = cl_list_tree_model=>align_auto.
*    item-font      = cl_list_tree_model=>item_font_prop.
*    SELECT SINGLE ttext FROM tobjt INTO item-text
*           WHERE langu  = sy-langu
*           AND   object = usr07-objct.
*    APPEND item TO item_table.

    DATA: lv_parprof LIKE wa_prof-parprof,
          it_all_comp LIKE TABLE OF wa_prof WITH HEADER LINE,
          it_roles TYPE TABLE OF agr_prof WITH HEADER LINE,
          wa_role LIKE LINE OF it_roles,
          it_roletxt TYPE TABLE OF agr_texts,
          wa_roletxt LIKE LINE OF it_roletxt,
          it_usr11 TYPE TABLE OF usr11,
          wa_usr11 LIKE LINE OF it_usr11.

    SORT it_all BY "prof_bukrs
      profile parprof bname.
    DELETE ADJACENT DUPLICATES FROM it_all COMPARING
      profile parprof bname.

    SORT it_all BY profile bname.
    it_all_comp[] = it_all[].

    SORT it_all_comp BY profile parprof.
    DELETE ADJACENT DUPLICATES FROM it_all_comp COMPARING profile parprof.

    DATA: last_prof LIKE it_all-profile,
          last_comp LIKE it_all-profile,
          lv_bname LIKE wa_prof-bname,
          lv_curr_bname LIKE wa_prof-bname,    "bname for 'at' processing
          lv_node_key LIKE node-node_key,
          lv_text LIKE item-text,
          lv_expander(1).

    CLEAR: last_prof, last_comp, lv_bname.
```

```
    IF it_all[] IS NOT INITIAL.

      SELECT * FROM agr_prof INTO TABLE it_roles FOR ALL ENTRIES IN it_all WHERE
 profile = it_all-profile.

      IF it_roles[] IS NOT INITIAL.
        SELECT * FROM agr_texts INTO TABLE it_roletxt
          FOR ALL ENTRIES IN it_roles
          WHERE agr_name = it_roles-agr_name
            AND spras IN ('EN', 'DE').
      ENDIF.

*Fill profile texts
      SELECT * FROM usr11 INTO TABLE it_usr11
        FOR ALL ENTRIES IN it_all
        WHERE profn = it_all-profile
              AND   aktps = aktivated
              AND   langu = sy-langu.

    ENDIF.

    LOOP AT it_all INTO wa_prof.

      MOVE: wa_prof-parprof TO lv_parprof,
            wa_prof-bname to lv_curr_bname.

      AT NEW profile.
        CLEAR lv_text.

        READ TABLE it_roles INTO wa_role WITH KEY profile = wa_prof-
profile langu = 'EN'.
        IF sy-subrc <> 0.
          READ TABLE it_roles INTO wa_role WITH KEY profile = wa_prof-
profile langu = 'DE'.
          IF sy-subrc = 0.
            READ TABLE it_roletxt WITH KEY agr_name = wa_role-
agr_name spras = 'D' INTO wa_roletxt.
          ENDIF.
        ELSE.
          READ TABLE it_roletxt WITH KEY agr_name = wa_role-
agr_name spras = 'E' INTO wa_roletxt.
        ENDIF.

        IF wa_roletxt IS NOT INITIAL.
          lv_text = 'Role'.

          ADD 1 TO g_node_key_num.
          CLEAR node.
          node-node_key    = g_node_key_num.
          node-relatkey    = usr07-objct.
          node-relatship   = cl_list_tree_model=>relat_last_child.
          node-n_image     = 'BNONE'.
          node-isfolder    = 'X'.
          if lv_curr_bname is not INITIAL.
            node-expander    = lv_expander.
          else.
            node-expander    = ''.
          endif.
*     The node is marked with a '+', although it has no children. When the
```

```
*     user clicks on the + to open the node, the event expand_nc is fired.
*     The programmer can add the children of the node within the event
*     handler of the expand_nc event (see callback handle_expand_nc).
          APPEND node TO node_table.

          CLEAR wa_node_table.
          wa_node_table-node_key       = node-node_key.
          wa_node_table-isfolder       = node-isfolder.
          wa_node_table-expander       = node-expander.
          wa_node_table-objct          = usr07-objct.
          wa_node_table-agr_name       = wa_role-agr_name.
          wa_node_table-profile        = wa_prof-profile.
          INSERT wa_node_table INTO TABLE gt_nodes.

*         Items for composite profiles
          CLEAR item.
          item-node_key   = node-node_key.
          item-item_name  = '01'.
          item-class      = cl_list_tree_model=>item_class_text.
          item-alignment  = cl_list_tree_model=>align_auto.
          item-font       = cl_list_tree_model=>item_font_prop.
          item-text       = lv_text.
          APPEND item TO item_table.

          CLEAR item.
          item-node_key   = node-node_key.
          item-item_name  = '02'.
          item-class      = cl_list_tree_model=>item_class_text.
          item-length     = 30.
          item-font       = cl_list_tree_model=>item_font_fixed.
          item-usebgcolor = usebgcolor. " light grey background
          item-style      = 9. "COL_KEY
          if wa_role-agr_name is NOT INITIAL.
            item-text       = wa_role-agr_name.
          else.
            item-text       = wa_prof-profile.
          endif.
          APPEND item TO item_table.

          CLEAR item.
          item-node_key   = node-node_key.
          item-item_name  = '03'.
          item-class      = cl_list_tree_model=>item_class_text.
          item-alignment  = cl_list_tree_model=>align_auto.
          item-font       = cl_list_tree_model=>item_font_prop.
          item-text       = wa_roletxt-text.
          APPEND item TO item_table.
          CLEAR: wa_role,
                 wa_roletxt.
        ELSE.
          IF lv_parprof IS NOT INITIAL.      "profile is part of a composite pro
file
            lv_text       = 'Composite Profile'.            "#EC *
          ELSE.
            lv_text       = 'Profile'(083).                 "#EC *
            lv_expander = p_user.            "Clear this if you don't want to sh
ow the users
          ENDIF.
```

```
          ADD 1 TO g_node_key_num.
          CLEAR node.
          node-node_key   = g_node_key_num.
          node-relatkey   = usr07-objct.
          node-relatship  = cl_list_tree_model=>relat_last_child.
          node-n_image    = 'BNONE'.
          node-isfolder   = 'X'.
          if lv_curr_bname is not INITIAL.
            node-expander   = lv_expander.
          else.
            node-expander   = ''.
          endif.
*     The node is marked with a '+', although it has no children. When the
*     user clicks on the + to open the node, the event expand_nc is fired.
*     The programmer can add the children of the node within the event
*     handler of the expand_nc event (see callback handle_expand_nc).
          APPEND node TO node_table.

          CLEAR wa_node_table.
          wa_node_table-node_key       = node-node_key.
          wa_node_table-isfolder       = node-isfolder.
          wa_node_table-expander       = node-expander.
          wa_node_table-objct          = usr07-objct.
          wa_node_table-profile        = wa_prof-profile.
          INSERT wa_node_table INTO TABLE gt_nodes.

*         Items for composite profiles
          CLEAR item.
          item-node_key   = node-node_key.
          item-item_name  = '01'.
          item-class      = cl_list_tree_model=>item_class_text.
          item-alignment  = cl_list_tree_model=>align_auto.
          item-font       = cl_list_tree_model=>item_font_prop.
          item-text       = lv_text.
          APPEND item TO item_table.

          CLEAR item.
          item-node_key   = node-node_key.
          item-item_name  = '02'.
          item-class      = cl_list_tree_model=>item_class_text.
          item-length     = 12.
          item-font       = cl_list_tree_model=>item_font_fixed.
          item-usebgcolor = usebgcolor. " light grey background
          item-style      = 9. "COL_KEY
          item-text       = wa_prof-profile.
          APPEND item TO item_table.

          CLEAR item.
          item-node_key   = node-node_key.
          item-item_name  = '03'.
          item-class      = cl_list_tree_model=>item_class_text.
          item-alignment  = cl_list_tree_model=>align_auto.
          item-font       = cl_list_tree_model=>item_font_prop.

*profile short text
          READ TABLE it_usr11 INTO wa_usr11
            WITH KEY profn = wa_prof-profile
                     aktps = aktivated
                     langu = sy-langu.
```

```
          IF sy-subrc <> 0.
            CLEAR item-text.
          ELSE.
            MOVE wa_usr11-ptext TO item-text.
          ENDIF.
          APPEND item TO item_table.
*          ENDIF.
          CLEAR: lv_parprof.
        ENDIF.
      ENDAT.

      IF wa_prof-
parprof IS NOT INITIAL      "profile is part of a composite profile
        AND wa_prof-profile <> last_comp.
        LOOP AT it_all_comp WHERE profile = wa_prof-profile.
          IF last_prof <> it_all_comp-parprof.
            ADD 1 TO g_node_key_role.
            CLEAR node.
            node-node_key   = g_node_key_role.
            node-relatkey   = g_node_key_num.
            node-relatship  = cl_list_tree_model=>relat_last_child.
            node-n_image    = 'BNONE'.
            node-isfolder   = 'X'.
            APPEND node TO node_table.

            CLEAR l_wa_node_table.
            l_wa_node_table-node_key       = node-node_key.
            l_wa_node_table-isfolder       = node-isfolder.
            l_wa_node_table-expander       = node-expander.
            l_wa_node_table-profile        = it_all_comp-parprof.
            INSERT l_wa_node_table INTO TABLE gt_nodes.

            CLEAR item.
            item-node_key   = node-node_key.
            item-item_name  = '01'.
            item-class      = cl_list_tree_model=>item_class_text.
            item-alignment  = cl_list_tree_model=>align_auto.
            item-font       = cl_list_tree_model=>item_font_prop.
            item-text       = 'contains profile'.           "#EC *
            APPEND item TO item_table.

            CLEAR item.
            item-node_key   = node-node_key.
            item-item_name  = '02'.
            item-class      = cl_list_tree_model=>item_class_text.
            item-length     = 12.
            item-font       = cl_list_tree_model=>item_font_fixed.
            item-usebgcolor = usebgcolor. " light grey background
            item-style      = 10.
            item-text       = it_all_comp-parprof.
            APPEND item TO item_table.

            CLEAR item.
            item-node_key   = node-node_key.
            item-item_name  = '03'.
            item-class      = cl_list_tree_model=>item_class_text.
            item-length     = 60.
            item-font       = cl_list_tree_model=>item_font_prop.
```

```
*profile short text
            READ TABLE it_usr11 INTO wa_usr11
              WITH KEY profn = it_all_comp-parprof
                       aktps = aktivated
                       langu = sy-langu.
            IF sy-subrc <> 0.
              CLEAR item-text.
            ELSE.
              MOVE wa_usr11-ptext TO item-text.
            ENDIF.
            APPEND item TO item_table.

            last_prof = it_all_comp-parprof.
          ENDIF.
        ENDLOOP.
        last_comp = wa_prof-profile.
        CLEAR: last_prof.
      ELSEIF wa_prof-profile = last_comp AND wa_prof-bname = lv_bname.
        CONTINUE.
      ENDIF.

*  *    user ids and details
      IF NOT wa_prof-bname IS INITIAL.

        ADD 1 TO g_node_key_role.
        CLEAR node.
        node-node_key   = g_node_key_role.
        node-relatkey   = g_node_key_num.
        node-relatship  = cl_list_tree_model=>relat_last_child.
        node-n_image    = 'BNONE'.
        node-isfolder   = 'X'.
*          node-expander   = 'X'.
        APPEND node TO node_table.

        CLEAR l_wa_node_table.
        l_wa_node_table-node_key       = node-node_key.
        l_wa_node_table-isfolder       = node-isfolder.
        l_wa_node_table-expander       = node-expander.
        l_wa_node_table-agr_name       = 'this is agr_name in bname'.
        INSERT l_wa_node_table INTO TABLE gt_nodes.

        CLEAR item.
        item-node_key   = node-node_key.
        item-item_name  = '01'.
        item-class      = cl_list_tree_model=>item_class_text.
        item-alignment  = cl_list_tree_model=>align_auto.
        item-font       = cl_list_tree_model=>item_font_prop.
        item-text       = 'User Id'.                          "#EC *
        APPEND item TO item_table.

        CLEAR item.
        item-node_key   = node-node_key.
        item-item_name  = '02'.
        item-class      = cl_list_tree_model=>item_class_text.
        item-alignment  = cl_list_tree_model=>align_auto.
        item-font       = cl_list_tree_model=>item_font_fixed.
        item-usebgcolor = usebgcolor. " light grey background
        item-style      = cl_list_tree_model=>style_emphasized.
        item-text       = wa_prof-bname.
```

```
        APPEND item TO item_table.

        CLEAR item.
        item-node_key   = node-node_key.
        item-item_name  = '03'.
        item-class      = cl_list_tree_model=>item_class_text.
        item-alignment  = cl_list_tree_model=>align_auto.
        item-font       = cl_list_tree_model=>item_font_prop.
        item-text       = wa_prof-name.
        APPEND item TO item_table.

      ELSE.
*        IF p_user = 'X'." and lv_repeat = 0.
*          ADD 1 TO g_node_key_role.
*          CLEAR node.
*          node-node_key   = g_node_key_role.
*          node-relatkey   = g_node_key_num.
*          node-relatship  = cl_list_tree_model=>relat_last_child.
*          node-n_image    = 'BNONE'.
*          node-isfolder   = ' '.
*          APPEND node TO node_table.
*
*          CLEAR l_wa_node_table.
*          l_wa_node_table-node_key       = node-node_key.
*          l_wa_node_table-isfolder       = node-isfolder.
*          l_wa_node_table-expander       = node-expander.
*          l_wa_node_table-agr_name        = 'this is agr_name in bname'.
*          INSERT l_wa_node_table INTO TABLE gt_nodes.
*
*          CLEAR item.
*          item-node_key   = node-node_key.
*          item-item_name  = '01'.
*          item-class      = cl_list_tree_model=>item_class_text.
*          item-alignment  = cl_list_tree_model=>align_auto.
*          item-font       = cl_list_tree_model=>item_font_prop.
*          item-text       = 'No user assigned!'.            "#EC *
*          APPEND item TO item_table.
*        ENDIF.
      ENDIF.
      lv_bname = wa_prof-bname.

    ENDLOOP.

    IF sy-subrc NE 0.

*   Node 'No authorization'
      CLEAR node.
      node-node_key   = 'NoAuth'.
      node-relatkey   = usr07-
objct. "'Root2'.                                "#EC NOTEXT
      node-relatship  = cl_list_tree_model=>relat_last_child.
      node-n_image    = 'BNONE'.
      node-isfolder   = ' '.
      APPEND node TO node_table.

      CLEAR item.
      item-node_key   = node-node_key.
      item-item_name  = '01'.
      item-class      = cl_list_tree_model=>item_class_text.
```

```
      item-alignment  = cl_list_tree_model=>align_auto.
      item-font       = cl_list_tree_model=>item_font_prop.
      item-text       = 'No Profiles available'(016).       "#EC *
      APPEND item TO item_table.

    ENDIF.

  ENDIF.

* ---------------------------------------------------------------------
* Folders with keys 'Root3' and 'Root4': HR-Trace
* ---------------------------------------------------------------------
  DATA l_item_text TYPE scrpcha72.

* Check HR-Trace

  CALL METHOD cl_hr_system_authority_trace=>is_trace_available "XMK...
    EXPORTING
      uname = g_bname
    EXCEPTIONS
      no_trace_available = 1
      OTHERS             = 2.
  IF sy-subrc = 0.

    CALL FUNCTION 'SAPGUI_PROGRESS_INDICATOR'
      EXPORTING
*       PERCENTAGE       = 0
        text             = 'Reading HR trace ...'(030)      "#EC *
              .

    FIELD-SYMBOLS: <trace> TYPE LINE OF hrtb_trace1,
                   <hyper> TYPE LINE OF hrtb_trace2.

    DATA: lt_trace TYPE hrtb_trace1,
          lt_hyper TYPE hrtb_trace2.

*   Read trace file for user
    CALL METHOD cl_hr_system_authority_trace=>trace_file_read
      EXPORTING
        uname              = g_bname
      IMPORTING
        trace_tab          = lt_trace
        trace_hyper_tab    = lt_hyper
      EXCEPTIONS
        no_trace_available = 1
        OTHERS             = 2.
    IF sy-subrc = 0.

*     Trace output: Step 1 Structural checks (VIEW)

      LOOP AT lt_trace ASSIGNING <trace>.

        AT FIRST.
          CLEAR node.
          node-node_key = 'Root3'.                          "#EC NOTEXT
          node-relatkey  = ' '.
          " Special case: A root node has no parent node
          node-relatship = cl_list_tree_model=>relat_next_sibling.
          node-hidden    = ' '.        " The node is visible,
```

```
    node-disabled  = ' '.          " selectable,
    node-isfolder  = 'X'.          " a folder,
    node-expander  = space.        " and a subtree.
    node-n_image   = 'BNONE'.      " Folder-Symbol in state "closed"
    node-exp_image = 'BNONE'.      " Folder-Symbol in state "open"
    APPEND node TO node_table.

    CLEAR wa_node_table.
    wa_node_table-node_key       = node-node_key.
    wa_node_table-isfolder       = node-isfolder.
    wa_node_table-expander       = node-expander.
    INSERT wa_node_table INTO TABLE gt_nodes.

    CLEAR item.
    item-node_key   = node-node_key.
    item-item_name  = '01'.
    item-class      = cl_list_tree_model=>item_class_text.
    " the with of the item is adjusted to its content (text)
    item-alignment  = cl_list_tree_model=>align_auto.
    " use proportional font for the item
    item-font       = cl_list_tree_model=>item_font_prop.
    item-text       =
      'Failed HR Structure Authorizations'(t01).      "#EC *
    APPEND item TO item_table.
  ENDAT.

  ADD 1 TO g_node_key_num.
  CLEAR node.
  node-node_key   = g_node_key_num.
  node-relatkey   = 'Root3'.                          "#EC NOTEXT
  node-relatship  = cl_list_tree_model=>relat_last_child.
  node-hidden     = ' '.  " The node is visible,
  node-disabled   = ' '.  " selectable,
  node-isfolder   = 'X'.  " a folder.
  node-n_image    = 'BNONE'.  "no icon
  node-expander   = ' '.
  APPEND node TO node_table.

  CLEAR wa_node_table.
  wa_node_table-node_key       = node-node_key.
  wa_node_table-isfolder       = node-isfolder.
  wa_node_table-expander       = node-expander.
  INSERT wa_node_table INTO TABLE gt_nodes.

  CLEAR item.
  item-node_key   = node-node_key.
  item-item_name  = '01'.
  item-class      = cl_list_tree_model=>item_class_text.
  item-alignment  = cl_list_tree_model=>align_auto.
  item-font       = cl_list_tree_model=>item_font_prop.
  item-text       = 'Date'(032).                      "#EC *
  APPEND item TO item_table.

  CLEAR item.
  item-node_key   = node-node_key.
  item-item_name  = '02'.
  item-class      = cl_list_tree_model=>item_class_text.
  item-length     = 19.
  item-font       = cl_list_tree_model=>item_font_fixed.
```

```
        item-usebgcolor = usebgcolor. " light grey background
*       item-style      = cl_list_tree_model=>style_emphasized_a.
        WRITE  <trace>-timestamp TIME ZONE sy-zonlo
          TO l_item_text.
        item-text       =  l_item_text.
        APPEND item TO item_table.

        CLEAR item.
        item-node_key   = node-node_key.
        item-item_name  = '03'.
        item-class      = cl_list_tree_model=>item_class_text.
        item-alignment  = cl_list_tree_model=>align_auto.
        item-font       = cl_list_tree_model=>item_font_prop.
        item-text       = 'Plan Version'(033).              "#EC *
        APPEND item TO item_table.

        CLEAR item.
        item-node_key   = node-node_key.
        item-item_name  = '04'.
        item-class      = cl_list_tree_model=>item_class_text.
        item-length     = 2.
        item-font       = cl_list_tree_model=>item_font_fixed.
        item-usebgcolor = usebgcolor. " light grey background
*       item-style      = cl_list_tree_model=>style_emphasized_a.
        item-text = <trace>-plvar.
        APPEND item TO item_table.

*       Item: otype
        CLEAR item.
        item-node_key   = node-node_key.
        item-item_name  = '05'.
        item-class      = cl_list_tree_model=>item_class_text.
        item-alignment  = cl_list_tree_model=>align_auto.
        item-font       = cl_list_tree_model=>item_font_prop.
        item-text       = 'Object Type'(034).               "#EC *
        APPEND item TO item_table.

        CLEAR item.
        item-node_key   = node-node_key.
        item-item_name  = '06'.
        item-class      = cl_list_tree_model=>item_class_text.
        item-length     = 2.
        item-font       = cl_list_tree_model=>item_font_fixed.
        item-usebgcolor = usebgcolor. " light grey background
*       item-style      = cl_list_tree_model=>style_emphasized_a.
        item-text = <trace>-otype.
        APPEND item TO item_table.

*       Item: objid
        CLEAR item.
        item-node_key   = node-node_key.
        item-item_name  = '07'.
        item-class      = cl_list_tree_model=>item_class_text.
        item-alignment  = cl_list_tree_model=>align_auto.
        item-font       = cl_list_tree_model=>item_font_prop.
        item-text       = 'Object ID'(035).                 "#EC *
        APPEND item TO item_table.

        CLEAR item.
```

```
        item-node_key    = node-node_key.
        item-item_name   = '08'.
        item-class       = cl_list_tree_model=>item_class_text.
        item-length      = 8.
        item-font        = cl_list_tree_model=>item_font_fixed.
        item-usebgcolor = usebgcolor. " light grey background
*       item-style       = cl_list_tree_model=>style_emphasized_a.
        item-text = <trace>-objid.
        APPEND item TO item_table.

*       Item: begda
        IF NOT <trace>-begda IS INITIAL.
          CLEAR item.
          item-node_key    = node-node_key.
          item-item_name   = '09'.
          item-class       = cl_list_tree_model=>item_class_text.
          item-alignment   = cl_list_tree_model=>align_auto.
          item-font        = cl_list_tree_model=>item_font_prop.
          item-text        = 'Start Date'(036).              "#EC *
          APPEND item TO item_table.

          CLEAR item.
          item-node_key    = node-node_key.
          item-item_name   = '10'.
          item-class       = cl_list_tree_model=>item_class_text.
          item-length      = 10.
          item-font        = cl_list_tree_model=>item_font_fixed.
          item-usebgcolor = usebgcolor. " light grey background
*         item-style       = cl_list_tree_model=>style_emphasized_a.
          WRITE <trace>-begda DD/MM/YYYY TO l_item_text.
          item-text =  l_item_text.
          APPEND item TO item_table.
        ENDIF.

*       Item: endda
        IF NOT <trace>-endda IS INITIAL.
          CLEAR item.
          item-node_key    = node-node_key.
          item-item_name   = '11'.
          item-class       = cl_list_tree_model=>item_class_text.
          item-alignment   = cl_list_tree_model=>align_auto.
          item-font        = cl_list_tree_model=>item_font_prop.
          item-text        = 'End Date'(037).                "#EC *
          APPEND item TO item_table.

          CLEAR item.
          item-node_key    = node-node_key.
          item-item_name   = '12'.
          item-class       = cl_list_tree_model=>item_class_text.
          item-length      = 10.
          item-font        = cl_list_tree_model=>item_font_fixed.
          item-usebgcolor = usebgcolor. " light grey background
*         item-style       = cl_list_tree_model=>style_emphasized_a.
          WRITE <trace>-endda DD/MM/YYYY TO l_item_text.
          item-text        = l_item_text.
          APPEND item TO item_table.
        ENDIF.

*       Item: action
```

```
        CLEAR item.
        item-node_key    = node-node_key.
        item-item_name   = '13'.
        item-class       = cl_list_tree_model=>item_class_text.
        item-alignment   = cl_list_tree_model=>align_auto.
        item-font        = cl_list_tree_model=>item_font_prop.
        item-text        = 'Action'(038).                    "#EC *
        APPEND item TO item_table.

        CLEAR item.
        item-node_key    = node-node_key.
        item-item_name   = '14'.
        item-class       = cl_list_tree_model=>item_class_text.
        item-length      = 12.
        item-font        = cl_list_tree_model=>item_font_fixed.
        item-usebgcolor = usebgcolor. " light grey background
*       item-style       = cl_list_tree_model=>style_emphasized_a.
        item-text = <trace>-action.
        APPEND item TO item_table.
      ENDLOOP.

*     Trace output: Step 2 Object creation checks (HYPERVIEW)
      LOOP AT lt_hyper ASSIGNING <hyper>.

        AT FIRST.
          CLEAR node.
          node-node_key = 'Root4'.                          "#EC NOTEXT
          node-relatkey  = ' '.
          " Special case: A root node has no parent node
          node-relatship = cl_list_tree_model=>relat_next_sibling.
          node-hidden    = ' '.         " The node is visible,
          node-disabled  = ' '.         " selectable,
          node-isfolder  = 'X'.         " a folder,
          node-expander  = space.       " and a subtree.
          node-n_image   = 'BNONE'.     " Folder-Symbol in state "closed"
          node-exp_image = 'BNONE'.     " Folder-Symbol in state "open"
          APPEND node TO node_table.

          CLEAR wa_node_table.
          wa_node_table-node_key       = node-node_key.
          wa_node_table-isfolder       = node-isfolder.
          wa_node_table-expander       = node-expander.
          INSERT wa_node_table INTO TABLE gt_nodes.

          CLEAR item.
          item-node_key    = node-node_key.
          item-item_name   = '01'.
          item-class       = cl_list_tree_model=>item_class_text.
          " the with of the item is adjusted to its content (text)
          item-alignment   = cl_list_tree_model=>align_auto.
          " use proportional font for the item
          item-font        = cl_list_tree_model=>item_font_prop.
          item-text        = 'HR Trace: Checks when Creating'(t02). "#EC *
          APPEND item TO item_table.
        ENDAT.

        ADD 1 TO g_node_key_num.
        CLEAR node.
        node-node_key    = g_node_key_num.
```

```
        node-relatkey   = 'Root4'.                          "#EC NOTEXT
        node-relatship  = cl_list_tree_model=>relat_last_child.
        node-hidden     = ' '.   " The node is visible,
        node-disabled   = ' '.   " selectable,
        node-isfolder   = 'X'.   " a folder.
        node-n_image    = 'BNONE'.  "no icon
        APPEND node TO node_table.

        CLEAR wa_node_table.
        wa_node_table-node_key       = node-node_key.
        wa_node_table-isfolder       = node-isfolder.
        wa_node_table-expander       = node-expander.
        INSERT wa_node_table INTO TABLE gt_nodes.

*       Item: timestamp
        CLEAR item.
        item-node_key   = node-node_key.
        item-item_name  = '01'.
        item-class      = cl_list_tree_model=>item_class_text.
        item-alignment  = cl_list_tree_model=>align_auto.
        item-font       = cl_list_tree_model=>item_font_prop.
        item-text       = 'Date'(032).                       "#EC *
        APPEND item TO item_table.

        CLEAR item.
        item-node_key   = node-node_key.
        item-item_name  = '02'.
        item-class      = cl_list_tree_model=>item_class_text.
        item-length     = 19.
        item-font       = cl_list_tree_model=>item_font_fixed.
        item-usebgcolor = usebgcolor. " light grey background
*       item-style      = cl_list_tree_model=>style_emphasized_a.
        WRITE  <hyper>-timestamp TIME ZONE sy-zonlo
          TO l_item_text.
        item-text       = l_item_text.
        APPEND item TO item_table.

*       Item: plvar
        CLEAR item.
        item-node_key   = node-node_key.
        item-item_name  = '03'.
        item-class      = cl_list_tree_model=>item_class_text.
        item-alignment  = cl_list_tree_model=>align_auto.
        item-font       = cl_list_tree_model=>item_font_prop.
        item-text       = 'Plan Version'(033).               "#EC *
        APPEND item TO item_table.

        CLEAR item.
        item-node_key   = node-node_key.
        item-item_name  = '04'.
        item-class      = cl_list_tree_model=>item_class_text.
        item-length     = 2.
        item-font       = cl_list_tree_model=>item_font_fixed.
        item-usebgcolor = usebgcolor. " light grey background
*       item-style      = cl_list_tree_model=>style_emphasized_a.
        item-text = <hyper>-plvar.
        APPEND item TO item_table.

*       Item: otype
```

```
        CLEAR item.
        item-node_key    = node-node_key.
        item-item_name   = '05'.
        item-class       = cl_list_tree_model=>item_class_text.
        item-alignment   = cl_list_tree_model=>align_auto.
        item-font        = cl_list_tree_model=>item_font_prop.
        item-text        = 'Object Type'(034).              "#EC *
        APPEND item TO item_table.

        CLEAR item.
        item-node_key    = node-node_key.
        item-item_name   = '06'.
        item-class       = cl_list_tree_model=>item_class_text.
        item-length      = 2.
        item-font        = cl_list_tree_model=>item_font_fixed.
        item-usebgcolor = usebgcolor. " light grey background
*       item-style       = cl_list_tree_model=>style_emphasized_a.
        item-text = <hyper>-otype.
        APPEND item TO item_table.

*       Item: relation
        CLEAR item.
        item-node_key    = node-node_key.
        item-item_name   = '07'.
        item-class       = cl_list_tree_model=>item_class_text.
        item-alignment   = cl_list_tree_model=>align_auto.
        item-font        = cl_list_tree_model=>item_font_prop.
        item-text        = 'Relation'(039).                 "#EC *
        APPEND item TO item_table.

        CLEAR item.
        item-node_key    = node-node_key.
        item-item_name   = '08'.
        item-class       = cl_list_tree_model=>item_class_text.
        item-length      = 4.
        item-font        = cl_list_tree_model=>item_font_fixed.
        item-usebgcolor = usebgcolor. " light grey background
*       item-style       = cl_list_tree_model=>style_emphasized_a.
        l_item_text+0(1) = <hyper>-rsign.
        l_item_text+1(3) = <hyper>-relat.
        item-text         = l_item_text.
        APPEND item TO item_table.

*       Item: sclas
        CLEAR item.
        item-node_key    = node-node_key.
        item-item_name   = '09'.
        item-class       = cl_list_tree_model=>item_class_text.
        item-alignment   = cl_list_tree_model=>align_auto.
        item-font        = cl_list_tree_model=>item_font_prop.
        item-text        = 'Object Type'(034).              "#EC *
        APPEND item TO item_table.

        CLEAR item.
        item-node_key    = node-node_key.
        item-item_name   = '10'.
        item-class       = cl_list_tree_model=>item_class_text.
        item-length      = 2.
        item-font        = cl_list_tree_model=>item_font_fixed.
```

```
        item-usebgcolor = usebgcolor. " light grey background
*       item-style      = cl_list_tree_model=>style_emphasized_a.
        item-text = <hyper>-sclas.
        APPEND item TO item_table.
      ENDLOOP.

    ENDIF.
  ENDIF.

ENDFORM.                               " build_node_and_item_table

*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_0100  INPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE user_command_0100 INPUT.
  PERFORM user_command_0100.
ENDMODULE.                    "user_command_0100 INPUT
*
FORM user_command_0100.
  DATA: return_code      TYPE i,
        sel_node_key     TYPE tm_nodekey,
        collapse_subtree TYPE as4flag.

  DATA p_tcode LIKE tstc-tcode.

* CL_GUI_CFW=>DISPATCH must be called if events are registered
* that trigger PAI
* this method calls the event handler method of an event
  CALL METHOD cl_gui_cfw=>dispatch
    IMPORTING
      return_code = return_code.
  IF return_code <> cl_gui_cfw=>rc_noevent.
    " a control event occured => exit PAI
    CLEAR g_ok_code.
    EXIT.
  ENDIF.

  CASE g_ok_code.

*   Back
    WHEN 'BACK' OR 'RW' OR 'CANC'. " Finish program
      IF NOT g_custom_container IS INITIAL.
        " destroy tree container (detroys contained tree control, too)
        CALL METHOD g_custom_container->free
          EXCEPTIONS
            cntl_system_error = 1
            cntl_error        = 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.
        CLEAR g_custom_container.
        CLEAR g_tree.
      ENDIF.
      LEAVE PROGRAM.

*   Other user / auth. object
```

```
    WHEN 'USER'.
      AUTHORITY-CHECK OBJECT 'S_USER_GRP'
           ID 'CLASS' DUMMY
           ID 'ACTVT' FIELD '03'.
      IF sy-subrc <> 0.
        MESSAGE e495. " Keine Berechtigung zum Anzeigen
      ENDIF.

      IF NOT g_custom_container IS INITIAL.
        " destroy tree container (detroys contained tree control, too)
        CALL METHOD g_custom_container->free
          EXCEPTIONS
            cntl_system_error = 1
            cntl_error        = 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.
        CLEAR g_custom_container.
        CLEAR g_tree.
        CLEAR gt_nodes[].
      ENDIF.
      PERFORM display_other_user.

*   Expand subtree
    WHEN 'EXPAND'.
      DATA isfolder(1).
      PERFORM get_selected_node USING sel_node_key isfolder.
      CHECK ( NOT sel_node_key IS INITIAL ) AND isfolder = 'X'.

      CALL FUNCTION 'SAPGUI_PROGRESS_INDICATOR'
        EXPORTING
*         PERCENTAGE       = 0
          text             = 'Display...'(028)              "#EC *
          .

*     Load not expanded subtrees
      DATA: sel_node_table LIKE LINE OF gt_nodes,
            wa_node_table LIKE LINE OF gt_nodes,
            node_table    TYPE treemlnota,
            item_table    TYPE treemlitac.

      CLEAR sel_node_table.
      READ TABLE gt_nodes INTO sel_node_table
        WITH KEY node_key = sel_node_key. "Sorted table

      CLEAR: node_table[], item_table[].

      LOOP AT gt_nodes INTO wa_node_table
        WHERE isfolder = 'X'
          AND expander = 'X'
          AND NOT objct IS INITIAL
          AND NOT auth  IS INITIAL.

        CHECK sel_node_key = usr07-objct "'Root2'
           OR (    wa_node_table-objct = sel_node_table-objct
               AND sel_node_table-auth IS INITIAL )
           OR (    wa_node_table-objct = sel_node_table-objct
               AND wa_node_table-profile  = sel_node_table-profile  )
```

```
              .

        CLEAR wa_node_table-expander.
        MODIFY gt_nodes FROM wa_node_table.

      ENDLOOP.

      IF NOT node_table[] IS INITIAL.
        CALL METHOD g_tree->add_nodes
          EXPORTING
            node_table          = node_table
          EXCEPTIONS
            error_in_node_table = 1
            OTHERS              = 2.

        CALL METHOD g_tree->add_items
          EXPORTING
            item_table          = item_table
          EXCEPTIONS
            error_in_item_table = 1
            OTHERS              = 2.

        CALL METHOD g_tree->create_tree_control
          EXPORTING
            parent                      = g_custom_container
          EXCEPTIONS
            lifetime_error              = 1
            cntl_system_error           = 2
            create_error                = 3
            failed                      = 4
            tree_control_already_created = 5
            OTHERS                      = 6.

*       ENDIF.
      ENDIF.

*     Expand subtree
      CALL METHOD g_tree->expand_node
        EXPORTING
          node_key            = sel_node_key
*         LEVEL_COUNT         =
          expand_subtree      = 'X'
        EXCEPTIONS
          node_not_found      = 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.

*   Collapse subtree
    WHEN 'COLLAPSE'.
      PERFORM get_selected_node USING sel_node_key isfolder.
      CHECK ( NOT sel_node_key IS INITIAL ) AND isfolder = 'X'.

      CALL METHOD g_tree->collapse_node
        EXPORTING
          node_key         = sel_node_key
          collapse_subtree = 'X'
```

```
        EXCEPTIONS
          node_not_found   = 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.

    WHEN 'RESET'.
*     Reset user buffer
      CALL FUNCTION 'SUSR_USER_BUFFER_AFTER_CHANGE'
        EXPORTING
          username = g_bname
        IMPORTING
          subrc    = return_code.
      IF return_code = 0.
        MESSAGE s055(01). "Benutzerpuffer erfolgreich zurückgesetzt.
      ELSE.
        MESSAGE s056(01). "Fehler beim Rücksetzen der Benutzerpuffer.
      ENDIF.

    WHEN 'PRI'.
      PERFORM print_tree USING 'X'.

    WHEN 'TRAN'.
      p_tcode = 'SU56'.

      CALL FUNCTION 'AUTHORITY_CHECK_TCODE'
        EXPORTING
          tcode  = p_tcode
        EXCEPTIONS
          ok     = 1
          not_ok = 2
          OTHERS = 3.
      IF sy-subrc = 1.
        CALL TRANSACTION p_tcode.
      ELSE.
*       No authorisation rights to run the transaction
        MESSAGE e172(00) WITH p_tcode.
      ENDIF.

    WHEN '%SC'.
      PERFORM find.

    WHEN '%SC+'.
      PERFORM find_next.

  ENDCASE.

* CAUTION: clear ok code!
  CLEAR g_ok_code.
ENDFORM.                    " USER_COMMAND_0100  INPUT

*&---------------------------------------------------------------------*
*&      Form  get_selected_node
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->L_SEL_NODE_text
```

```
*       -->L_IS_FOLDERtext
*---------------------------------------------------------------------*
FORM get_selected_node
  USING l_sel_node_key TYPE tm_nodekey
        l_isfolder.

  DATA: wa_node_table TYPE gt_nodes_struct.

  CLEAR: l_sel_node_key, l_isfolder.

  CALL METHOD g_tree->get_selected_item
    IMPORTING
      node_key          = l_sel_node_key
*     ITEM_NAME         =
    EXCEPTIONS
      failed            = 1
      cntl_system_error = 2
      no_item_selection = 3
      OTHERS            = 4.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.
  IF l_sel_node_key IS INITIAL.
    CALL METHOD g_tree->get_selected_node
      IMPORTING
        node_key                  = l_sel_node_key
      EXCEPTIONS
        failed                    = 1
        single_node_selection_only = 2
        cntl_system_error         = 3
        OTHERS                    = 4.
    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
              WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.
  ENDIF.

*  CHECK NOT l_sel_node_key IS INITIAL.
  IF l_sel_node_key IS INITIAL.
    MESSAGE i141(1x).
    EXIT.
  ENDIF.

  READ TABLE gt_nodes INTO wa_node_table
    WITH KEY node_key = l_sel_node_key.
  IF sy-subrc = 0 AND wa_node_table-isfolder = 'X'.
    l_isfolder = 'X'.
  ENDIF.

ENDFORM.                    "get_selected_node

*&---------------------------------------------------------------------*
*&      Form  print_tree
*&---------------------------------------------------------------------*

FORM print_tree USING p_preview TYPE as4flag.

  DATA: l_title    TYPE string,
```

```
        l_datum    TYPE char10,
        temp0(50)  TYPE c,
        temp1(50)  TYPE c.

  l_title = 'Evaluation of the Last Failed'(040).           "#EC *

  temp0 = 'Authorization Checks for User'(045).             "#EC *

  temp1 = g_bname.

  CONCATENATE l_title temp0 temp1 INTO l_title SEPARATED BY space.

  CALL METHOD g_tree->print_tree
    EXPORTING
      all_nodes            = 'X'
      title                = l_title
      preview              = p_preview
    EXCEPTIONS
      control_not_existing = 1
      control_dead         = 2
      cntl_system_error    = 3
      failed               = 4
      OTHERS               = 5.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.

ENDFORM.                    " print_tree
*&---------------------------------------------------------------------*
*&      Form  find
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
FORM find.
  DATA:
    l_result_type              TYPE i,
    l_result_item_key_table    TYPE treemiks,
    l_result_item_key          TYPE treemikey,
    l_result_expander_node_key TYPE tm_nodekey,
    l_find_string              TYPE string,
    l_pattern                  TYPE as4flag,
    l_canceled                 TYPE as4flag.

  CLEAR: g_result_item_key_table,
         g_find_string.

  CALL FUNCTION 'TREEM_SHOW_FIND_POPUP'
    IMPORTING
      find_string = l_find_string
      pattern     = l_pattern
      canceled    = l_canceled.

  IF l_canceled IS INITIAL.
    CALL METHOD g_tree->find_all
      EXPORTING
```

```
        search_string            = l_find_string
*         ITEM_NAME_TABLE         =
        pattern_search           = l_pattern
*         START_NODE              =
*         STOP_AT_EXPANDER_NODE   =
      IMPORTING
        result_type              = l_result_type
        result_item_key_table    = l_result_item_key_table
        result_expander_node_key = l_result_expander_node_key
      EXCEPTIONS
        start_node_not_found     = 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.
      EXIT.
    ENDIF.

    READ TABLE l_result_item_key_table INTO l_result_item_key INDEX 1.
    IF sy-subrc EQ 0.
      g_result_item_key_table = l_result_item_key_table.
      g_find_string = l_find_string.
      CALL METHOD g_tree->select_item
        EXPORTING
          node_key          = l_result_item_key-node_key
          item_name         = l_result_item_key-item_name
        EXCEPTIONS
          no_item_selection = 1
          node_not_found    = 2
          item_not_found    = 3
          OTHERS            = 4.
      IF sy-subrc <> 0.
*        MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*                   WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
      ELSE.
        DELETE g_result_item_key_table INDEX 1.
      ENDIF.
    ELSE.
      MESSAGE s469(s#) WITH l_find_string.
    ENDIF.

  ENDIF.
ENDFORM.                    " find_string
*&---------------------------------------------------------------------*
*&      Form  find_next
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
FORM find_next .

  DATA l_result_item_key        TYPE treemikey.

  READ TABLE g_result_item_key_table INTO l_result_item_key INDEX 1.
  IF sy-subrc EQ 0.
    CALL METHOD g_tree->select_item
```

```
      EXPORTING
        node_key          = l_result_item_key-node_key
        item_name         = l_result_item_key-item_name
      EXCEPTIONS
        no_item_selection = 1
        node_not_found    = 2
        item_not_found    = 3
        OTHERS            = 4.
    IF sy-subrc <> 0.
*       MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*                  WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
    ELSE.
      DELETE g_result_item_key_table INDEX 1.
    ENDIF.
  ELSE.
    MESSAGE s186(oo) WITH g_find_string.
  ENDIF.

ENDFORM.                    " find_next
*&---------------------------------------------------------------------*
*&      Form  header_data
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->P_NODE_TABLE  text
*      -->P_ITEM_TABLE  text
*      -->P_I  text
*----------------------------------------------------------------------*
FORM header_data   USING node_table      TYPE treemlnota
                         item_table      TYPE treemlitac.


  DATA: l_text1 TYPE tm_itemtxt,
        l_text2 TYPE tm_itemtxt,
        l_text3 TYPE tm_itemtxt,
        l_text4 TYPE tm_itemtxt,
        l_item_table      TYPE treemlitac,
        l_node_table      TYPE treemlnota,
        l_nodes_to_expand TYPE treemnotab,
        l_node TYPE treemlnodt,
        l_item TYPE treemliten.

  DATA: date_long  TYPE c LENGTH 10,
        time_long  TYPE c LENGTH 10.

  l_node-node_key  = 'Blank'(047).                          "#EC *
  l_node-hidden    = ' '.  " The node is visible,
  l_node-disabled  = ' '.  " selectable,
  l_node-isfolder  = 'X'.  " a folder.
  l_node-n_image   = 'BNONE'.
  l_node-expander  = space.      " and a subtree.
  l_node-exp_image = 'BNONE'.   " Folder in state "open"
  APPEND l_node TO l_node_table.

  CLEAR l_item.
  l_item-node_key   = l_node-node_key.
  l_item-item_name  = '01'.
  l_item-class      = cl_list_tree_model=>item_class_text.
  l_item-length     = 95.
```

```
  l_item-font       = cl_list_tree_model=>item_font_default.
  l_item-text       =
 '------------------------------------------------------------------------'
.
  l_text1 =
'------------------------------------------------------------------------'
   .
  l_text2 =
   '--------------------------------'
  .
  CONCATENATE l_item-text l_text1 l_text2 INTO l_item-text.
  APPEND l_item TO l_item_table.
  node_table = l_node_table.
  item_table = l_item_table.

  CLEAR l_text1.
  CLEAR l_text2.
  CLEAR l_text3.
  CLEAR l_text4.
  l_text1 = 'Instance'(049).                                  "#EC *
  l_text2 = sy-host.
  l_text3 = 'Profile parameter auth/new buffering'(075).      "#EC *
  l_text4 = g_buffer_method.
  l_node-node_key = 'Instance'(049).                          "#EC *

  PERFORM header USING l_node l_text1 l_text2 l_text3 l_text4
                 l_node_table l_item_table l_nodes_to_expand.

  node_table = l_node_table.
  item_table = l_item_table.

  CLEAR l_text1.
  CLEAR l_text2.
  CLEAR l_text3.
  CLEAR l_text4.
  l_text1 = 'Date'(032).                                      "#EC *
  WRITE sy-datum TO date_long.
  l_text2 = date_long.
  l_text3 = 'Time'(046).                                      "#EC *
  WRITE sy-uzeit TO time_long.
  l_text4 = time_long.
  l_node-node_key = 'Date'(032).                              "#EC *

  PERFORM header USING l_node l_text1 l_text2 l_text3 l_text4
                 l_node_table l_item_table l_nodes_to_expand.

  node_table = l_node_table.
  item_table = l_item_table.

  CLEAR l_text1.
  CLEAR l_text2.
  CLEAR l_text3.
  CLEAR l_text4.
  l_text1 = 'System'(078).                                    "#EC *
  l_text2 = sy-sysid.
  l_text3 = 'Client'(043).                                    "#EC *
  l_text4 = sy-mandt.
  l_node-node_key = 'System'(078).                            "#EC *
```

```
  PERFORM header USING l_node l_text1 l_text2 l_text3 l_text4
                  l_node_table l_item_table l_nodes_to_expand.

  node_table = l_node_table.
  item_table = l_item_table.

  CLEAR l_text1.
  CLEAR l_text2.
  CLEAR l_text3.
  CLEAR l_text4.
  l_text1 = 'User name'(080).                                  "#EC *
  l_text2 = g_bname.
  l_text3 = 'Authorization Object'(079).                       "#EC *
  l_text4 = g_objct.
  l_node-node_key = 'User'(048).                               "#EC *

  PERFORM header USING l_node l_text1 l_text2 l_text3 l_text4
                  l_node_table l_item_table l_nodes_to_expand.

  node_table = l_node_table.
  item_table = l_item_table.

ENDFORM.                    " header_data
*&---------------------------------------------------------------------*
*&      Form  header
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->P_L_NODE  text
*      -->P_L_TEXT1  text
*      -->P_L_TEXT2  text
*      -->P_L_TEXT3  text
*      -->P_L_TEXT4  text
*      -->P_L_NODE_TABLE  text
*      -->P_L_ITEM_TABLE  text
*      -->P_L_NODES_TO_EXPAND  text
*----------------------------------------------------------------------*
FORM header  USING    p_l_node TYPE treemlnodt
                      p_l_text1
                      p_l_text2
                      p_l_text3
                      p_l_text4
                      p_l_node_table TYPE treemlnota
                      p_l_item_table TYPE treemlitac
                      p_l_nodes_to_expand TYPE treemnotab.



  DATA: wa_node_table LIKE LINE OF gt_nodes,
        l_node TYPE treemlnodt,
        l_item TYPE treemliten.

  CLEAR l_node.
  l_node-node_key = p_l_node-node_key.                         "#EC NOTEXT
  l_node-hidden    = ' '.         " The node is visible,
  l_node-disabled  = ' '.         " selectable,
  l_node-isfolder  = 'X'.         " a folder,
  l_node-n_image   = 'BNONE'.
  l_node-expander  = space.       " and a subtree.
```

```
  l_node-exp_image = 'BNONE'.    " Folder-/ Leaf-Symbol in state "open"
  APPEND l_node TO p_l_node_table.


  CLEAR l_item.
  l_item-node_key    = l_node-node_key.
  l_item-item_name   = '01'.
  l_item-class       = cl_list_tree_model=>item_class_text.
  l_item-length      = 30.
  l_item-font        = cl_list_tree_model=>item_font_default.
  l_item-text        = p_l_text1.
  APPEND l_item TO p_l_item_table.


  CLEAR l_item.
  l_item-node_key    = l_node-node_key.
  l_item-item_name   = '02'.
  l_item-class       = cl_list_tree_model=>item_class_text.
  l_item-length      = 10.
  l_item-font        = cl_list_tree_model=>item_font_default.
  l_item-usebgcolor = usebgcolor. " light grey background
  IF NOT p_l_text2 IS INITIAL.
    l_item-style      = 8. "cl_list_tree_model=>style_emphasized_a.
    "COL_HEADING
  ENDIF.
  l_item-text        = p_l_text2.
  APPEND l_item TO p_l_item_table.

  IF NOT p_l_text3 IS INITIAL.
    CLEAR l_item.
    l_item-node_key    = l_node-node_key.
    l_item-item_name   = '03'.
    l_item-class       = cl_list_tree_model=>item_class_text.
    l_item-length      = 40.
    l_item-font        = cl_list_tree_model=>item_font_default.
    l_item-text        = p_l_text3.
    APPEND l_item TO p_l_item_table.

    CLEAR l_item.
    l_item-node_key    = l_node-node_key.
    l_item-item_name   = '04'.
    l_item-class       = cl_list_tree_model=>item_class_text.
    l_item-length      = 10.
    l_item-font        = cl_list_tree_model=>item_font_default.
    l_item-usebgcolor = usebgcolor. " light grey background
    l_item-style      = 8. "cl_list_tree_model=>style_emphasized_a.
    "COL_HEADING
    l_item-text        = p_l_text4.
    APPEND l_item TO p_l_item_table.
  ENDIF.

ENDFORM.                    " header

*&---------------------------------------------------------------------*
*&      Form  get_related_auth
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM get_related_auth TABLES lt_field_value STRUCTURE ls_field_value.
```

```
  REFRESH: it_profn, it_auth1, it_auth2, it_auth3, it_auth4, it_auth5,
           it_auth6, it_auth7, it_auth8, it_auth9, it_all, it_profiles,
           it_auth,  lt_ust04, it_auth10.

  CLEAR:   it_profn, it_auth1, it_auth2, it_auth3, it_auth4, it_auth5,
           it_auth6, it_auth7, it_auth8, it_auth9, it_all, it_profiles,
           lv_before, lv_after, it_auth, lt_ust04, ls_ust04, it_auth10.
* get all authorizations which fit objct and fiel*

  DATA: lv_field_count(2) TYPE n.
  SELECT DISTINCT * FROM ust12
                    INTO TABLE it_ust12
                    WHERE objct = usr07-objct AND
                          field IN (usr07-fiel0, usr07-fiel1, usr07-fiel2,
                                    usr07-fiel3, usr07-fiel4, usr07-fiel5,
                                    usr07-fiel6, usr07-fiel7, usr07-fiel8,
                                    usr07-fiel9).

* distinguish which authorizations fit p_value
  LOOP AT lt_field_value.
    lv_field_count = lv_field_count + 1.
    PERFORM dist_auth USING lv_field_count lt_field_value-field lt_field_value-
value.
    CASE lv_field_count.
      WHEN 1.
        SORT it_auth1 BY auth.
      WHEN 2.
        SORT it_auth2 BY auth.
      WHEN 3.
        SORT it_auth3 BY auth.
      WHEN 4.
        SORT it_auth4 BY auth.
      WHEN 5.
        SORT it_auth5 BY auth.
      WHEN 6.
        SORT it_auth6 BY auth.
      WHEN 7.
        SORT it_auth7 BY auth.
      WHEN 8.
        SORT it_auth8 BY auth.
      WHEN 9.
        SORT it_auth9 BY auth.
      WHEN 10.
        SORT it_auth10 BY auth.
      WHEN OTHERS.
    ENDCASE.
  ENDLOOP.

* compare authorizations with each other
* only authorizations with all selection criteria should be processed
  PERFORM comp_auth.

* which profiles correspond with authorizations
  PERFORM auth2prof.

  MOVE 1 TO level.

* is the profile within a composite profile
```

```
  DESCRIBE TABLE it_profn LINES lv_before.
  WHILE lv_before <> lv_after.
    IF lv_before <> lv_after AND lv_after <> 0.
      lv_before = lv_after.
    ENDIF.

    SELECT profn subprof FROM ust10c
      APPENDING CORRESPONDING FIELDS OF TABLE it_profn
                FOR ALL ENTRIES IN it_profn
                WHERE subprof = it_profn-profn AND aktps = 'A'.

    SORT it_profn BY profn auth subprof.
    DELETE ADJACENT DUPLICATES FROM it_profn COMPARING profn auth subprof.
    DESCRIBE TABLE it_profn LINES lv_after.
  ENDWHILE.

  DATA: lv_rc LIKE sy-subrc,
        lv_user_class LIKE usr02-class.

  IF it_profn[] IS NOT INITIAL.

    SELECT SINGLE class FROM usr02 INTO lv_user_class
      WHERE bname = usr07-bname.
    PERFORM auth_check(saplsuu0)
          USING obj_group lv_user_class space act_show lv_rc.
    IF lv_rc = 0.
      CALL FUNCTION 'SAPGUI_PROGRESS_INDICATOR'
      EXPORTING
*       PERCENTAGE       = 0
        text             = 'Reading assigned users...'.     "#EC *

      IF p_user = 'X'.
        SELECT DISTINCT
            t1~bname
            t1~profile
            t3~name_first AS vorna
            t3~name_last AS nachn
         FROM
            usr02 AS t2
          INNER JOIN ust04 AS t1 ON
            t2~bname = t1~bname
          INNER JOIN user_addrp AS t3 ON
             t2~bname = t3~bname
          INTO CORRESPONDING FIELDS OF TABLE lt_ust04
                                FOR ALL ENTRIES IN it_profn
                                WHERE profile = it_profn-profn.
      ELSE.
        SELECT DISTINCT
              t2~mandt
              t1~bname
              t1~profile
           FROM
              usr02 AS t2
            INNER JOIN ust04 AS t1 ON
              t2~bname = t1~bname
            INTO CORRESPONDING FIELDS OF TABLE lt_ust04
                                  FOR ALL ENTRIES IN it_profn
                                  WHERE profile = it_profn-profn.
      ENDIF.
```

```
    ELSE.
      MESSAGE ID '01' TYPE 'I' NUMBER '512' WITH lv_user_class.
    ENDIF.
  ENDIF.

  LOOP AT it_profn WHERE auth <> ''.
    PERFORM find_profiles USING it_profn-profn space level.
  ENDLOOP.

  it_all[] = it_profiles[].

ENDFORM.                    "get_related_auth

*---------------------------------------------------------------------*
*       Form  dist_auth
*---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
*      -->VALUE  text
*---------------------------------------------------------------------*
* distinguish which authorizations fit value
FORM dist_auth USING no
                     field
                     value.
  DATA: lv_len TYPE i,
        lv_len_counter TYPE i.
  DATA: lv_val0(10), lv_val1(10),
        lv_val2(10), lv_val3(10),
        lv_val4(10), lv_val5(10),
        lv_val6(10), lv_val7(10),
        lv_val8(10), lv_val9(10),
        lv_match_pattern.

  lv_len = STRLEN( value ).
  IF lv_len > 0.
    lv_len_counter = 0.
    DO lv_len TIMES.
      CASE lv_len_counter.
        WHEN 0.
          CONCATENATE value(sy-index) '*' INTO lv_val0.
        WHEN 1.
          CONCATENATE value(sy-index) '*' INTO lv_val1.
        WHEN 2.
          CONCATENATE value(sy-index) '*' INTO lv_val2.
        WHEN 3.
          CONCATENATE value(sy-index) '*' INTO lv_val3.
        WHEN 4.
          CONCATENATE value(sy-index) '*' INTO lv_val4.
        WHEN 5.
          CONCATENATE value(sy-index) '*' INTO lv_val5.
        WHEN 6.
          CONCATENATE value(sy-index) '*' INTO lv_val6.
        WHEN 7.
          CONCATENATE value(sy-index) '*' INTO lv_val7.
        WHEN 8.
          CONCATENATE value(sy-index) '*' INTO lv_val8.
        WHEN 9.
          CONCATENATE value(sy-index) '*' INTO lv_val9.
      ENDCASE.
```

```
      lv_len_counter = lv_len_counter + 1.
    ENDDO.
  ENDIF.

  LOOP AT it_ust12 WHERE field = field.
    IF value IS INITIAL. "in case the authorization was checked with "Dummy"
      PERFORM append_auth USING no it_ust12-auth
                                it_ust12-von
                                it_ust12-bis.
    ELSE.
      IF it_ust12-von CA '*' OR it_ust12-bis CA '*'.
        lv_match_pattern = 'X'.
      ELSE.
        lv_match_pattern = '' .
      ENDIF.

      IF it_ust12-bis = ''.
        IF it_ust12-von EQ value OR it_ust12-von = '*'.
          PERFORM append_auth USING no it_ust12-auth
                                     it_ust12-von
                                     ''.
        ELSE.
          IF lv_match_pattern = 'X' AND ( ( it_ust12-
von = lv_val0 AND lv_val0 IS NOT INITIAL )
             OR ( it_ust12-von = lv_val1 AND lv_val1 IS NOT INITIAL )
             OR ( it_ust12-von = lv_val2 AND lv_val2 IS NOT INITIAL )
             OR ( it_ust12-von = lv_val3 AND lv_val3 IS NOT INITIAL )
             OR ( it_ust12-von = lv_val4 AND lv_val4 IS NOT INITIAL )
             OR ( it_ust12-von = lv_val5 AND lv_val5 IS NOT INITIAL )
             OR ( it_ust12-von = lv_val6 AND lv_val6 IS NOT INITIAL )
             OR ( it_ust12-von = lv_val7 AND lv_val7 IS NOT INITIAL )
             OR ( it_ust12-von = lv_val8 AND lv_val8 IS NOT INITIAL )
             OR ( it_ust12-von = lv_val9 AND lv_val9 IS NOT INITIAL ) ).
             PERFORM append_auth USING no it_ust12-auth
                                        it_ust12-von
                                        ''.
          ENDIF.
        ENDIF.
      ELSE.
        IF value >= it_ust12-von AND value <= it_ust12-bis.
          PERFORM append_auth USING no it_ust12-auth
                                     it_ust12-von
                                     it_ust12-bis.
        ELSE.
          IF lv_match_pattern = 'X'
            AND ( ( ( lv_val0 >= it_ust12-von AND lv_val0 <= it_ust12-
bis ) AND lv_val0 IS NOT INITIAL )
             OR ( ( lv_val1 >= it_ust12-von AND lv_val1 <= it_ust12-
bis ) AND lv_val1 IS NOT INITIAL )
             OR ( ( lv_val2 >= it_ust12-von AND lv_val2 <= it_ust12-
bis ) AND lv_val2 IS NOT INITIAL )
             OR ( ( lv_val3 >= it_ust12-von AND lv_val3 <= it_ust12-
bis ) AND lv_val3 IS NOT INITIAL )
             OR ( ( lv_val4 >= it_ust12-von AND lv_val4 <= it_ust12-
bis ) AND lv_val4 IS NOT INITIAL )
             OR ( ( lv_val5 >= it_ust12-von AND lv_val5 <= it_ust12-
bis ) AND lv_val5 IS NOT INITIAL )
             OR ( ( lv_val6 >= it_ust12-von AND lv_val6 <= it_ust12-
bis ) AND lv_val6 IS NOT INITIAL )
```

```
            OR ( ( lv_val7 >= it_ust12-von AND lv_val7 <= it_ust12-
bis ) AND lv_val7 IS NOT INITIAL )
            OR ( ( lv_val8 >= it_ust12-von AND lv_val8 <= it_ust12-
bis ) AND lv_val8 IS NOT INITIAL )
            OR ( ( lv_val9 >= it_ust12-von AND lv_val9 <= it_ust12-
bis ) AND lv_val9 IS NOT INITIAL ) ).
            PERFORM append_auth USING no it_ust12-auth
                                      it_ust12-von
                                      it_ust12-bis.
          ENDIF.
        ENDIF.
      ENDIF.
    ENDIF.
    CLEAR wa_auth.
  ENDLOOP.
ENDFORM.                    " dist_auth

*---------------------------------------------------------------------*
*       Form  append_auth
*---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*---------------------------------------------------------------------*
FORM append_auth USING no auth
                       bis
                       von.
  MOVE: auth TO wa_auth-auth,
        bis TO wa_auth-bis,
        von TO wa_auth-von.
  CASE no.
    WHEN 1.
      APPEND wa_auth TO it_auth1.
    WHEN 2.
      APPEND wa_auth TO it_auth2.
    WHEN 3.
      APPEND wa_auth TO it_auth3.
    WHEN 4.
      APPEND wa_auth TO it_auth4.
    WHEN 5.
      APPEND wa_auth TO it_auth5.
    WHEN 6.
      APPEND wa_auth TO it_auth6.
    WHEN 7.
      APPEND wa_auth TO it_auth7.
    WHEN 8.
      APPEND wa_auth TO it_auth8.
    WHEN 9.
      APPEND wa_auth TO it_auth9.
    WHEN 10.
      APPEND wa_auth TO it_auth10.
  ENDCASE.
ENDFORM.                    " append_auth

*---------------------------------------------------------------------*
*       Form  comp_auth
*---------------------------------------------------------------------*
*       text
```

```
*---------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*---------------------------------------------------------------------*
* compare authorizations with each other
* only authorizations with all selection criteria should be processed
FORM comp_auth.
  DATA: rc LIKE sy-subrc.

  IF NOT usr07-fiel1 IS INITIAL.
    LOOP AT it_auth1.
      READ TABLE it_auth2 WITH KEY auth = it_auth1-auth
                    BINARY SEARCH.
      rc = sy-subrc.
      IF NOT usr07-fiel2 IS INITIAL.
        READ TABLE it_auth3 WITH KEY auth = it_auth1-auth
                      BINARY SEARCH.
        rc = rc + sy-subrc.
      ENDIF.
      IF NOT usr07-fiel3 IS INITIAL.
        READ TABLE it_auth4 WITH KEY auth = it_auth1-auth
                      BINARY SEARCH.
        rc = rc + sy-subrc.
      ENDIF.
      IF NOT usr07-fiel4 IS INITIAL.
        READ TABLE it_auth5 WITH KEY auth = it_auth1-auth
                      BINARY SEARCH.
        rc = rc + sy-subrc.
      ENDIF.
      IF NOT usr07-fiel5 IS INITIAL.
        READ TABLE it_auth6 WITH KEY auth = it_auth1-auth
                      BINARY SEARCH.
        rc = rc + sy-subrc.
      ENDIF.
      IF NOT usr07-fiel6 IS INITIAL.
        READ TABLE it_auth7 WITH KEY auth = it_auth1-auth
                      BINARY SEARCH.
        rc = rc + sy-subrc.
      ENDIF.
      IF NOT usr07-fiel7 IS INITIAL.
        READ TABLE it_auth8 WITH KEY auth = it_auth1-auth
                      BINARY SEARCH.
        rc = rc + sy-subrc.
      ENDIF.
      IF NOT usr07-fiel8 IS INITIAL.
        READ TABLE it_auth9 WITH KEY auth = it_auth1-auth
                      BINARY SEARCH.
        rc = rc + sy-subrc.
      ENDIF.
      IF NOT usr07-fiel9 IS INITIAL.
        READ TABLE it_auth10 WITH KEY auth = it_auth1-auth
                      BINARY SEARCH.
        rc = rc + sy-subrc.
      ENDIF.
      IF rc = 0.
        APPEND it_auth1 TO it_auth.
      ENDIF.
    ENDLOOP.
  ELSE.
```

```
    it_auth[] = it_auth1[].
  ENDIF.
ENDFORM.                    " comp_auth

*---------------------------------------------------------------------*
*       Form  auth2prof
*---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*---------------------------------------------------------------------*
* which profiles correspond with authorizations
FORM auth2prof.
  IF it_auth[] IS NOT INITIAL.


    DATA: lv_temp TYPE i.
    CLEAR lv_temp.


    SELECT DISTINCT profn auth FROM ust10s
                          INTO TABLE it_profn
                          FOR ALL ENTRIES IN it_auth
                          WHERE objct EQ usr07-objct
                          AND auth EQ it_auth-auth
                          AND aktps EQ 'A'.

    SELECT DISTINCT profn auth FROM ust10s
                          APPENDING TABLE it_profn
                          WHERE objct EQ usr07-objct
                          AND auth EQ '*'
                          AND aktps EQ 'A'.

    SORT it_profn BY profn auth.

    DELETE ADJACENT DUPLICATES FROM it_profn[] COMPARING profn.
  ENDIF.
ENDFORM.                                                    " auth2prof

*---------------------------------------------------------------------*
*       Form  FIND_PROFILES
*---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*---------------------------------------------------------------------*
FORM find_profiles USING profile parent level.
  DATA: BEGIN OF itab OCCURS 0,
          profn LIKE usr10-profn,
        END OF itab,
        new_level(2) TYPE n.

  REFRESH: itab.

* directly associated profiles
  LOOP AT lt_ust04 INTO ls_ust04
    WHERE profile = profile.
```

```
    MOVE: ls_ust04-bname TO wa_prof-bname,
          ls_ust04-profile TO wa_prof-profile,
          level TO wa_prof-level,
          parent TO wa_prof-parprof. ",
    CONCATENATE ls_ust04-vorna ls_ust04-nachn
        INTO wa_prof-name SEPARATED BY space.
    APPEND wa_prof TO it_profiles.
    CLEAR: ls_ust04, it_profiles.
  ENDLOOP.
* no entry found, but show the category
  IF sy-subrc NE 0.
    MOVE: level TO wa_prof-level,
          profile TO wa_prof-profile,
          'no user found' TO wa_prof-name,
          parent TO wa_prof-parprof. ",
    APPEND wa_prof TO it_profiles.
    CLEAR it_profiles.
  ENDIF.
  CLEAR wa_prof.

* is the profile within a composite profile
  LOOP AT it_profn WHERE subprof = profile.
    MOVE: it_profn-profn TO itab-profn. ",
    APPEND itab.
  ENDLOOP.
  IF sy-subrc = 0.
* search for users who have composite profiles of interest
    new_level = level + 1.
    LOOP AT itab.
      PERFORM find_profiles USING itab-profn it_profn-subprof new_level.
    ENDLOOP.
  ENDIF.
ENDFORM.                    " FIND_PROFILES
```

Further Possibilities

This program could be extended further depending on the specific customer needs.

E.g.

- group roles/profiles by company codes
- logging cability and the ability of re-executing the program based on the log for IT/Basis colleagues wouldn't always have to ask for the screen shot from the users to resolve their problem
- Connect this application to a GRC (or other) system for role/profile provisioning

We'd be interested to hear your feedback on this as comments to either this article or the blog that we intent to create in reference to this article.

Related Content

For more information, visit the ABAP homepage.

Copyright