Filtering BW any version (from 7.X to BW/4HANA) hierarchy datasource via abap program layer - Backend solution

<u>Note:</u> This real scenario solution can be applied any version of BW systems, from 7.X to BW/4HANA .

<u>Problem :</u>

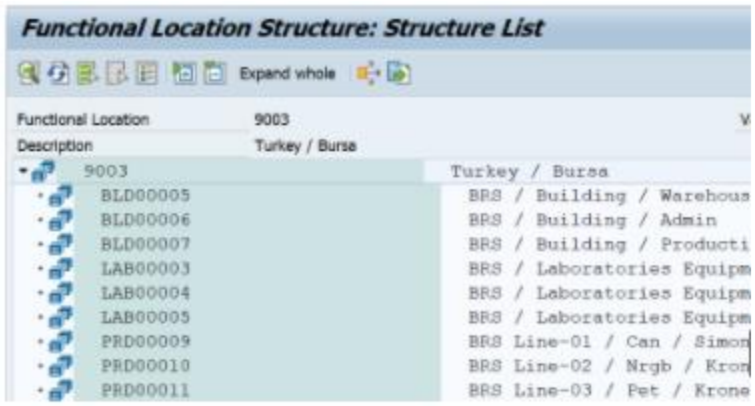
During loading Technical Location of the Hierarchy values (0FUNCT_LOC_HIER) BW datasource from ERP system to BW system , OS and DB system performance problems occurred and memory usage consumed Gigabytes in both in ERP system and BW system and both side gaves ST22 dumps during loading activities.

In this system, in daily routine loads , this datasource load time lasted more than 3 hours and gave dumps and consumed too much memory in size, both in OS and DB system.

0FUNCT_LOC_HIER datasource correspond to ERP system **IH01** transaction data output.

This DataSource delivers hierarchy relationships for functional locations.

<u>ERP IH01 screen sample snapshot:</u>



```
Functional Location   9003
Description           Turkey / Bursa
9003                  Turkey / Bursa
  BLD00005              BRS / Building / Warehous
  BLD00006              BRS / Building / Admin
  BLD00007              BRS / Building / Producti
  LAB00003              BRS / Laboratories Equipm
  LAB00004              BRS / Laboratories Equipm
  LAB00005              BRS / Laboratories Equipm
  PRD00009              BRS Line-01 / Can / Simon
  PRD00010              BRS Line-02 / Nrgb / Kron
  PRD00011              BRS Line-03 / Pet / Krone
```

<u>Identifying and investigating the source of the problem</u>

The performance problem encountered in system was first shared with the Basis team.

In the examinations, it was determined that there is a very high resource consumption on the database side, especially for RAM usage.

Due to high RAM consumption, programs running on the system started getting dump.

The program that received an error was identified in the examination carried out across the dumps.

It was determined that the program that received the error was SE37 program (PMEX_HIERARCHY_TRANSFER_IFLOT) which is running in the BW data source ((0FUNCT_LOC_HIER) backend.

Solution:

After negotiated with ERP consultants, obsolete 'functional location' hierarchy values were still in the ERP system and no need to load obsolete values to BW side. According to their feedback, 'functional location category' between A-Z range was enough.

Obsolote 'functional location' hierarchy node and leaf values that remained in the system caused this error.

From that point, I debugged PMEX_HIERARCHY_TRANSFER_IFLOT program and consume enhancement places in this program to filter FLTYP functional location category .

After applying filter on Enhancement, only active and necessary data was loaded again in seconds , and the system did not fail and did not gave dumps anymore. OS and DB values became stable.

Note: Datasource infopackage or datasource DTP has no filter for FLTYP. So, I filtered it in background, from program layer.

I am sharing Enhancement place (**ZZPMEX_HIERARCHY_TRANSFER_IFL**) at the end of this code below (Bold part).

SE37 function module standard abap program name: PMEX_HIERARCHY_TRANSFER_IFLOT

Enhancement name: ZZPMEX_HIERARCHY_TRANSFER_IFL

```
FUNCTION pmex hierarchy transfer iflot.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(I REQUNR) TYPE  SBIWA S INTERFACE-REQUNR OPTIONAL
*"     VALUE(I OSOURCE) TYPE  RSAOT OLTPSOURCE OPTIONAL
*"     VALUE(I MAXSIZE) TYPE  SBIWA S INTERFACE-MAXSIZE OPTIONAL
*"     VALUE(I INITFLAG) TYPE  SBIWA S INTERFACE-INITFLAG OPTIONAL
*"     VALUE(I S HIEBAS) TYPE  RSAP S HIEBAS OPTIONAL
*"     VALUE(I S HIEFLAG) TYPE  RSAP S HIEFLAG OPTIONAL
*"     VALUE(I_S_HIERSEL) TYPE  RSAP_S_HIER_LIST OPTIONAL
*"  EXPORTING
*"     VALUE(E PACKAGES) TYPE  SBIWA S INTERFACE-INITFLAG
*"     VALUE(E_S_HEADER) TYPE  TPLHIERSEL
*"  TABLES
*"      I T LANGU TYPE  SBIWA T LANGU OPTIONAL
*"      E T HIETEXT TYPE  RSAP T HIETEXT OPTIONAL
*"      E T HIENODE STRUCTURE  TPLHIENODE OPTIONAL
*"      E T FOLDERT TYPE  RSAP_T_FOLDERT OPTIONAL
*"      E_T_HIEINTV OPTIONAL
*"  EXCEPTIONS
*"      INVALID CHABASNM HCLASS
*"      INVALID HIERARCHY FLAG
*"      INVALID HIERARCHY SELECT
*"      LANGU NOT SUPPORTED
*"      HIERARCHY TAB NOT_FOUND
*"      APPLICATION ERROR
*"----------------------------------------------------------------------
  DATA:  ls_hienode            LIKE tplhienode,
         ls hietext            TYPE rsap s hietext,
         ls langu              TYPE sbiwa s langu,
         lt tplnr ma           TYPE pmex1 ty tplnr ma_tab,
         lt_ddtel_conv         LIKE dd04t OCCURS 0,
         ls ddtel conv         LIKE dd04t,
         l counter             TYPE i.
  STATICS: tmp e t hienode TYPE STANDARD TABLE OF TPLHIENODE, "1114454
           tmp maxsize     TYPE SBIWA S INTERFACE-MAXSIZE,     "1114454
*          l initial load  LIKE SY-datar.            "1114454  "1402485
           ls funct loc hier   TYPE TFUNCT LOC HIER,           "1402485
           ls funct loc hiert  TYPE TFUNCT LOC HIERT,          "1402485
           lv alternat label   TYPE SY-datar,                  "1402485
           lt dynamic select   TYPE RSAOT T DYNAMIC SELECT,    "1402485
           lv db selected      TYPE SY-datar,                  "1402485
           lv part hier        TYPE SY-datar.                  "1402485
  DATA: ls tmp e t hienode LIKE TPLHIENODE,                    "1114454
        l count            TYPE I.                             "1114454
  DATA: ls itobcust        TYPE ITOBCUST,                      "1402485
        YX                 TYPE C VALUE 'X',                   "1402485
        ls tmp iflot       TYPE IFLOT,                         "1402485
        l tmp tplnr from   TYPE ILOM STRNO,                    "1402485
        l tmp tplnr to     TYPE ILOM STRNO.                    "1402485
  DATA: BEGIN OF lt no tplma OCCURS 0,                         "1402485
          tplnr TYPE IFLOT-tplnr,                              "1402485
        END OF lt no tplma,                                    "1402485
        BEGIN OF lt topmost OCCURS 0,                          "1402485
          tplma TYPE IFLOT-tplnr,                              "1402485
        END OF lt topmost,                                     "1402485
        BEGIN OF lt tmp tplnr ma OCCURS 0,                     "1402485
          tplnr TYPE IFLOT-tplnr,                              "1402485
          tplma TYPE IFLOT-tplma,                              "1402485
        END OF lt tmp tplnr ma,                                "1402485
        ls tmp tplnr ma LIKE lt tmp tplnr ma.                  "1402485
  DATA: BEGIN OF ls part hier,                                 "1402485
          tplnr TYPE IFLOT-tplnr,                              "1402485
          tplma TYPE IFLOT-tplma,                              "1402485
        END OF ls part hier.                                   "1402485
  DATA: lt select TYPE SBIWA T SELECT,                         "1402485
        ls select TYPE SBIWA S SELECT.                         "1402485
  FIELD-SYMBOLS: <fs_tplma> LIKE lt_topmost.                   "1402485

*--- hierarchy will be read complete
* e packages = sbiwa c flag off.                               "1114454
  e_packages = sbiwa_c_flag_on.                                "1114454
  IF i initflag = SBIWA C FLAG ON.                            "1402485
* IF l initial load IS INITIAL.                     "1114454  "1402485
*   l initial load = 'X'.                           "1114454  "1402485
    tmp_maxsize    = i_maxsize.                               "1114454

*--- No single root node neccessary -> Root nodes have no parent
    l_counter = 0.

************************************************************************
*                                       BEGIN OF NOTE 1402485 Part 1 *
************************************************************************
*---Initialisation Step 1:
* ..Check, if entries exist in table TFUNCT_LOC_HIER
    SELECT SINGLE * FROM TFUNCT_LOC_HIER
      INTO ls funct loc hier
      WHERE hienm = i_s_hiersel-hienm.

    IF ls funct loc hier IS INITIAL.
*   ..Customizng table not maintained   ->old logic
      IF i s hiersel-hienm <> 'TPLH'.
        RAISE INVALID_HIERARCHY_SELECT.
      ENDIF.
    ELSE.
*   ..Customizimg table maintained    ->NEW LOGIC......................*
*-----Initialisation Step 2:
*   ..Check, if alternative labelling is on or not
      CALL FUNCTION 'ILOX_ITOBCUST_READ'
        IMPORTING
          E ITOBCUST     = ls_itobcust
        EXCEPTIONS
          NOT MAINTAINED = 1
          OTHERS         = 2.
      IF SY-subrc <> 0 OR ls_itobcust-cnvrt = ' '.
        lv alternat_label = SPACE.
      ELSE.
        lv alternat_label = YX.
      ENDIF.
*-----Initialisation Step 3:
*   ..If alternative labelling is active, convert TPLNR
      IF NOT lv_alternat_label IS INITIAL.
        CALL FUNCTION 'CONVERSION_EXIT_TPLNR_OUTPUT'
          EXPORTING
            INPUT  = ls_funct_loc_hier-tplnr_from
         IMPORTING
           OUTPUT  = l tmp tplnr from.
        CALL FUNCTION 'CONVERSION_EXIT_TPLNR_OUTPUT'
          EXPORTING
            INPUT  = ls_funct_loc_hier-tplnr_to
         IMPORTING
           OUTPUT  = l_tmp_tplnr_to.
      ELSE.
        l_tmp_tplnr_from = ls_funct_loc_hier-tplnr_from.
        l_tmp_tplnr_to   = ls_funct_loc_hier-tplnr_to.
      ENDIF.
*-----Initialisation Step 4:
*   ..Check plausibility of customized values...
*   ..and build table of selection criteria for WHERE clause
      IF NOT ls_funct_loc_hier-tplnr_from IS INITIAL
        AND NOT ls_funct_loc_hier-tplnr_to IS INITIAL.
*     ..'From' value must be lower than 'To' value
        IF l_tmp_tplnr_from > l_tmp_tplnr_to.
          RAISE INVALID_HIERARCHY_SELECT.
        ENDIF.
*     ..Both FunctLocs must be topmost FunctLocs
        SELECT SINGLE * FROM IFLOT INTO ls_tmp_iflot
          WHERE tplnr = ls_funct_loc_hier-tplnr_from
          AND   tplma = SPACE.
        IF SY-subrc IS INITIAL.
          SELECT SINGLE * FROM IFLOT INTO ls_tmp_iflot
            WHERE tplnr = ls_funct_loc_hier-tplnr_to
            AND   tplma = SPACE.
          IF NOT SY-subrc IS INITIAL.
            RAISE INVALID_HIERARCHY_SELECT.
          ENDIF.
        ELSE.
          RAISE INVALID_HIERARCHY_SELECT.
        ENDIF.
        ls_select-sign    = 'I'.
        ls_select-option  = 'BT'.
        IF lv_alternat_label IS INITIAL.
          ls_select-fieldnm = 'TPLNR'.
          ls_select-low   = ls_funct_loc_hier-tplnr_from.
          ls_select-high  = ls_funct_loc_hier-tplnr_to.
        ELSE.
          ls_select-fieldnm = 'STRNO'.
          ls_select-low   = l_tmp_tplnr_from.
          ls_select-high  = l_tmp_tplnr_to.
        ENDIF.
        APPEND ls_select TO lt_select.
      ELSEIF ls_funct_loc_hier-tplnr_from IS INITIAL
        AND NOT ls_funct_loc_hier-tplnr_to IS INITIAL.
*     ..Use same behaviour than in RSA3 ->no extraction
        RAISE INVALID_HIERARCHY_SELECT.
      ELSEIF NOT ls_funct_loc_hier-tplnr_from IS INITIAL
        AND ls_funct_loc_hier-tplnr_to IS INITIAL.
*     ..FunctLoc must be either a topmost FunctLoc
*       or any node within a hierarchy
        SELECT SINGLE * FROM IFLOT INTO ls_tmp_iflot
          WHERE tplnr = ls_funct_loc_hier-tplnr_from.
        IF SY-subrc IS INITIAL.
          IF ls_tmp_iflot-tplma IS INITIAL.
*         ..'FROM' value is a topmost FUNCTLOC
            ls_select-sign    = 'I'.
            ls_select-option  = 'EQ'.
            IF lv_alternat_label IS INITIAL.
```

```
              ls_select-fieldnm = 'TPLNR'.
              ls_select-low   = ls_funct_loc_hier-tplnr_from.
            ELSE.
              ls_select-fieldnm = 'STRNO'.
              ls_select-low   = l_tmp_tplnr_from.
            ENDIF.
            APPEND ls_select TO lt_select.
          ELSE.
*         ..'FROM' value is any FunctLoc within any hierarchy
            lv_part_hier = YX.
          ENDIF.
        ENDIF.
      ELSE.
*     ..Neither 'FROM' nor 'TO' values exist,
*       so check, if other selection criteria exist.
        IF ls_funct_loc_hier-tplnr_single IS INITIAL.
          IF ls_funct_loc_hier-swerk IS INITIAL AND
            ls_funct_loc_hier-fltyp IS INITIAL AND
            ls_funct_loc_hier-tplkz IS INITIAL.
            RAISE INVALID_HIERARCHY_SELECT.
          ENDIF.
        ENDIF.
      ENDIF.
*   ..Build table of selection criteria for WHERE clause
      IF NOT ls_funct_loc_hier-swerk IS INITIAL.
        ls_select-fieldnm = 'SWERK'.
        ls_select-sign    = 'I'.
        ls_select-option  = 'EQ'.
        ls_select-low     = ls_funct_loc_hier-swerk.
        APPEND ls_select TO lt_select.
      ENDIF.
      IF NOT ls_funct_loc_hier-fltyp IS INITIAL.
        ls_select-fieldnm = 'FLTYP'.
        ls_select-sign    = 'I'.
        ls_select-option  = 'EQ'.
        ls_select-low     = ls_funct_loc_hier-fltyp.
        APPEND ls_select TO lt_select.
      ENDIF.
      IF NOT ls_funct_loc_hier-tplkz IS INITIAL.
        ls_select-fieldnm = 'TPLKZ'.
        ls_select-sign    = 'I'.
        ls_select-option  = 'EQ'.
        ls_select-low     = ls_funct_loc_hier-tplkz.
        APPEND ls_select TO lt_select.
      ENDIF.
*-----Initialisation Step 5:
*   ..Build dynamic WHERE clause for SELECT statements
      CALL FUNCTION 'RSAN_FILL_DYNAMICAL_SELECT'
        EXPORTING
          I_T_SELECT                = lt_select
        IMPORTING
          E_T_DYNAMIC_SELECT        = lt_dynamic_select
        EXCEPTIONS
```

```
          INVALID SELECTION CRITERIA = 1
          OTHERS                     = 2.
      IF NOT SY-subrc IS INITIAL.
        RAISE INVALID_HIERARCHY_SELECT.
      ENDIF.
    ENDIF.

  ELSE.

* ..Extraction and Transfer to BW...
    IF lv db selected IS INITIAL.
*   ..Set flag because of nonrecurring selection and building
*     of the desired hierarchies
      lv db selected = YX.
*   ..How to select the desired data depends on table TFUNCT LOC_HIER.
*     ->If TFUNCT LOC HIER has no entries, use the old logic,
*       otherwise the new one.
      IF ls funct loc_hier IS INITIAL.
*     ..Old logic !
*       That means, transfer A L L hierarchies as 1 big hierarchy to BW
************************************************************************
*                                          END OF NOTE 1402485 Part 1 *
************************************************************************
*---    read all functionl locations
        SELECT tplnr tplma FROM iflot INTO TABLE lt_tplnr_ma. "#EC CI_NOWHERE
        IF sy-subrc IS INITIAL.
          SORT lt tplnr ma BY tplma tplnr DESCENDING.
*---      build hierarchy
*         PERFORM insert children f03 USING e t hienode[]      "1114454
          PERFORM insert children f03 USING tmp e t hienode[]  "1114454
                                            lt_tplnr_ma
                                            ls hienode-nodeid
                                            ls hienode-tplnr
                                            ls_hienode-tlevel
                                            ls hienode-childid
                                            l_counter.
        ENDIF.

*-----   get text of data element in several languages
        CALL FUNCTION 'DD_DTEL_GET'
          EXPORTING
            roll_name   = 'PM_IFLOT_HIER_TEXT'
          TABLES
            dd04t_tab_a = lt_ddtel_conv.

*-----   get Hierarchy descriptions in the required languages
        LOOP AT i t langu INTO ls langu.
          READ TABLE lt ddtel conv INTO ls ddtel conv
               WITH KEY ddlanguage = ls_langu-langu.
          IF sy-subrc = 0.
            TRANSLATE ls ddtel conv-scrtext s TO UPPER CASE.  "#EC SYNTCHAR
            TRANSLATE ls ddtel conv-scrtext m TO UPPER CASE.  "#EC SYNTCHAR
            TRANSLATE ls_ddtel_conv-scrtext_l TO UPPER CASE.  "#EC SYNTCHAR
            ls hietext-langu = ls langu-langu.
            ls_hietext-txtsh = ls_ddtel_conv-scrtext_m.
            ls hietext-txtmd = ls ddtel conv-scrtext m.
            ls hietext-txtlg = ls ddtel conv-scrtext_l.
            APPEND ls_hietext TO e_t_hietext.
          ENDIF.
        ENDLOOP.

*       SORT e t hienode BY nodeid.                            "1114454
        SORT tmp e t hienode[] BY nodeid.                      "1114454
*       EXIT.                                       "1114454   "1402485
************************************************************************
*                                        BEGIN OF NOTE 1402485 Part 2 *
************************************************************************
      ELSE.
*     ..New logic !
*     ..Select all FunctLocs, which do not have a superior FunctLoc.
*       Those FunctLocs are either...
*       ->the topmost FunctLoc of a hierarchy
*         or
*       ->a SINGLE FunctLoc, that means no parent, no child
        IF lv part hier IS INITIAL.
*       ..A single hierarchiy or a range of hierarchies is desired
          IF lv alternat label IS INITIAL.
*         ..Alternative labelling is not active   ->use table IFLOT
            SELECT tplnr FROM iflot hier
              INTO TABLE lt_no_tplma
              WHERE (lt dynamic_select)
              GROUP BY tplnr.
            IF lt no tplma[] IS INITIAL.
              RAISE INVALID_HIERARCHY_SELECT.
            ENDIF.
          ELSE.
*         ..Alternative labelling is active   ->use table IFLOS
            SELECT tplnr FROM iflos hier
              INTO TABLE lt no tplma
              WHERE (lt dynamic_select)
              GROUP BY tplnr.
            IF lt no tplma[] IS INITIAL.
              RAISE INVALID_HIERARCHY_SELECT.
            ENDIF.
          ENDIF.
*         ..SELECT all FunctLocs, which are the topmosts of a hierarchy
          SELECT tplma FROM IFLOT INTO TABLE lt_topmost
            FOR ALL ENTRIES IN lt_no_tplma
            WHERE tplma = lt no tplma-tplnr.
          IF NOT lt topmost[] IS INITIAL.
            SORT lt_topmost BY tplma.
          ENDIF.
          IF NOT ls funct loc hier-tplnr_single IS INITIAL.
*         ..Extract only SINGLEs
*           ->that means FunctLocs without any assignment
            LOOP AT lt_no_tplma INTO ls_tmp_tplnr_ma-tplnr.
```

```
              READ TABLE lt_topmost
                WITH KEY tplma = ls_tmp_tplnr_ma-tplnr
                BINARY SEARCH
                TRANSPORTING NO FIELDS.
              IF NOT SY-subrc IS INITIAL.
                APPEND ls_tmp_tplnr_ma TO lt_tplnr_ma.
              ENDIF.
            ENDLOOP.
            REFRESH: lt_topmost,
                     lt_no_tplma.
            CLEAR:   lt_topmost,
                     lt_no_tplma.
          ELSE.
*         ..Delete table lt_no_tplma->no more needed
            REFRESH lt_no_tplma.
            CLEAR lt_no_tplma.
*         ..Select all hierarchy-members of the topmost FunctLocs
            LOOP AT lt_topmost ASSIGNING <fs_tplma>.
              MOVE <fs_tplma> TO ls_tmp_tplnr_ma-tplnr.
*           ..Insert 1st hierarchy level
              APPEND ls_tmp_tplnr_ma TO lt_tplnr_ma.
*           ..Determine corresponding childs on 2nd level
              SELECT tplnr tplma FROM IFLOT
                INTO TABLE lt_tmp_tplnr_ma
                WHERE tplma = <fs_tplma>.
*           ..Insert childs of 2nd hierarchy-level
              APPEND LINES OF lt_tmp_tplnr_ma TO lt_tplnr_ma.
              WHILE SY-subrc IS INITIAL.
*             ..Determine corresponding childs on 3rd to n-th level
                SELECT tplnr tplma FROM IFLOT
                  INTO TABLE lt_tmp_tplnr_ma
                  FOR ALL ENTRIES IN lt_tmp_tplnr_ma
                  WHERE tplma = lt_tmp_tplnr_ma-tplnr.
                IF SY-subrc IS INITIAL.
*               ..Insert childs of 3rd to n-th hierarchy-level
                  APPEND LINES OF lt_tmp_tplnr_ma TO lt_tplnr_ma.
                ENDIF.
              ENDWHILE.
            ENDLOOP.
            REFRESH lt_topmost.
            CLEAR   lt_topmost.
          ENDIF.
        ELSE.
*       ..FunctLoc is not the topmost, but any node within a hierarchy
          MOVE ls_funct_loc_hier-tplnr_from TO ls_part_hier-tplnr.
*       ..Insert 1st level of partial hierarchy
          APPEND ls_part_hier TO lt_tplnr_ma.
*       ..Determine corresponding childs on 2nd level
          SELECT tplnr tplma FROM IFLOT INTO TABLE lt_tmp_tplnr_ma
            WHERE tplma = ls_funct_loc_hier-tplnr_from.
          IF SY-subrc IS INITIAL.
*         ..Insert childs of 2nd level of partial hierarchy
            APPEND LINES OF lt_tmp_tplnr_ma TO lt_tplnr_ma.
            WHILE SY-subrc IS INITIAL.
*           ..Determine corresponding childs on 3rd to n-th level
              SELECT tplnr tplma FROM IFLOT INTO TABLE lt_tmp_tplnr_ma
                FOR ALL ENTRIES IN lt tmp tplnr ma
                WHERE tplma = lt tmp tplnr_ma-tplnr.
              IF SY-subrc IS INITIAL.
*             ..Insert childs of 3rd to n-th level of partial hierarchy
                APPEND LINES OF lt_tmp_tplnr_ma TO lt_tplnr_ma.
              ENDIF.
            ENDWHILE.
          ENDIF.
        ENDIF.

        IF NOT lt tplnr ma[] IS INITIAL.
          SORT lt tplnr ma BY tplma tplnr DESCENDING.
*       ..Build the hierarchy
          PERFORM insert_children_f03 USING tmp_e_t_hienode[]
                                            lt tplnr ma
                                            ls hienode-nodeid
                                            ls_hienode-tplnr
                                            ls hienode-tlevel
                                            ls hienode-childid
                                            l_counter.
        ENDIF.

        SORT tmp_e_t_hienode[] BY nodeid.

        SELECT SINGLE * FROM TFUNCT_LOC_HIERT
          INTO ls funct loc hiert
          WHERE hienm = ls funct loc hier-hienm.
        ls_hietext-langu = ls_funct_loc_hiert-spras.
        ls hietext-txtsh = ls funct loc hiert-txtsh.
        ls hietext-txtmd = ls funct loc hiert-txtmd.
        ls_hietext-txtlg = ls_funct_loc_hiert-txtlg.
        APPEND ls_hietext TO e_t_hietext.

      ENDIF.
    ENDIF.
* ..Now transfer the hierarchy package-wise to BW
***********************************************************************
*                                          END OF NOTE 1402485 Part 1 *
***********************************************************************
    LOOP AT tmp e t hienode INTO ls tmp e t hienode.           "1114454
      l count = l count + 1.                                   "1114454
      IF l count <= tmp maxsize.                               "1114454
        DELETE tmp e t hienode INDEX 1.                        "1114454
        APPEND ls tmp e t hienode TO e t hienode.              "1114454
      ELSE.                                                    "1114454
        EXIT.                                                  "1114454
      ENDIF.                                                   "1114454
    ENDLOOP.                                                   "1114454

    IF tmp_e_t_hienode[] IS INITIAL.                           "1114454
```

```
      e packages = sbiwa c flag off.                                     "1114454
    ENDIF.                                                               "1114454
    ADD 1 TO G_COUNTER_DATAPAKID.                                        "1114454

*--- build Header
    IF ls_funct_loc_hier IS INITIAL.                                     "1402485
      e s header-hienm  = 'TPLH'.
    ELSE.                                                                "1402485
      e s header-hienm = ls funct loc hier-hienm.                        "1402485
    ENDIF.                                                               "1402485
    e_s_header-hclass = 'TPLH'.

  ENDIF.                                                                 "1402485

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""""""""""""""""$"$\SE:(1) Function Module PMEX HIERARCHY TRANSFER IFLOT, End

                   A
*$*$-Start: (1)---------------------------------------------------------------
------------------$*$*
ENHANCEMENT 2  ZZPMEX_HIERARCHY_TRANSFER_IFL.    "active version

  IF i_initflag = SBIWA_C_FLAG_ON.

     lt_select = value #( ( fieldnm = 'FLTYP' sign = 'I' option = 'BT' low =
'A' high = 'Z' )
                          ).


      CALL FUNCTION 'RSAN_FILL_DYNAMICAL_SELECT'
        EXPORTING
          I_T_SELECT                = lt_select
        IMPORTING
          E_T_DYNAMIC_SELECT        = lt_dynamic_select
        EXCEPTIONS
          INVALID_SELECTION_CRITERIA = 1
          OTHERS                    = 2.
  endif.

ENDENHANCEMENT.
*$*$-End:   (1)---------------------------------------------------------------
------------------$*$*
ENDFUNCTION.
```

Thank you.

Cihan Ekin