Class: ZCL_ZAS_MSTR_VER_DPC_EXT

Status: Active
Inh. from: ZCL_ZAS_MSTR_VER_DPC

Attributes

Description: Data Provider Secondary Class
Instantiation: Public
Not final

Not released
Fixed pt.arithmetic
Category: General Object Type
Package: ZSD
Original lang.: EN

Created by: SG0303539
Created on: 04/04/2019
Last changed on: 05/15/2019

Methods

Redefined Methods

/IWBEP/IF_MGW_APPL_SRV_RUNTIME~EXECUTE_ACTION

```
  METHOD /iwbep/if_mgw_appl_srv_runtime~execute_action.
**TRY.
*CALL METHOD SUPER->/IWBEP/IF_MGW_APPL_SRV_RUNTIME~EXECUTE_ACTION
**  EXPORTING
**    iv_action_name          =
**    it_parameter            =
**    io_tech_request_context =
**  IMPORTING
**    er_data                 =
*    .
** CATCH /iwbep/cx_mgw_busi_exception .
** CATCH /iwbep/cx_mgw_tech_exception .
**ENDTRY.
**Constant declaration
    CONSTANTS : con_nr_range   TYPE nrnr VALUE '01',
                con_nr_object  TYPE nrobj VALUE 'ZASRPVRSN',
                con_item_categ TYPE postp VALUE 'N',
                con_comp_qty   TYPE kmpmg_bi VALUE '1'.
    CONSTANTS: con_matype    TYPE bapimatdoa-matl_type VALUE 'ZRPK',
               con_indsr     TYPE bapimatdoa-ind_sector VALUE 'S',
               con_bom_usage TYPE csap_mbom-stlan VALUE '5',
               con_uom_ea    TYPE meins VALUE 'EA',
               con_item_cat  TYPE mtpos_mara VALUE 'Z001',
               con_lang      TYPE spras  VALUE 'E',
               con_plant     TYPE werks_d VALUE '1001'.
**Data declaration
    DATA: lt_key_tab TYPE        /iwbep/t_mgw_name_value_pair,
          ls_key_tab TYPE        /iwbep/s_mgw_name_value_pair.
    DATA: lo_function_import   TYPE REF TO object.
```

```
    DATA: ls_status            TYPE zcl_zas_mstr_ver_mpc=>ts_status.
    DATA: lt_rp_vrsn_mast_upd TYPE TABLE OF zas_rp_vrsn_mast,
          ls_rp_vrsn_mast_upd TYPE          zas_rp_vrsn_mast.
    DATA: ls_rp_vrsn_mast      TYPE zas_rp_vrsn_mast.
**Variable declaration
    DATA: lv_ga_date    TYPE zas_ga_date,
          lv_n_rank     TYPE zas_n_rank,
          lv_rp_desc    TYPE zas_rp_desc,
          lv_rp_version TYPE zas_rp_version,
          lv_serv_flag  TYPE zas_serv_flag,
          lv_product    TYPE matnr,
          lv_rpdesc     TYPE zas_rp_desc,
          lv_rpdesc_to  TYPE zas_rp_desc,
          lv_number     TYPE zas_rp_version_int,
          lv_count      TYPE i,
          lv_date       TYPE char10,
          lv_bsegment   TYPE char15.
    DATA: ls_stk2 TYPE stko_api02,
          lt_stp2 TYPE TABLE OF stpo_api02,
          ls_stp2 TYPE stpo_api02.
    DATA: lt_dep_source TYPE TABLE OF dep_source,
          ls_dep_source TYPE dep_source.
    DATA: lv_warning TYPE capiflag-flwarning.
*    DATA: lv_rel_pkg_atinn    TYPE atinn,
*          lv_rel_pkg          TYPE atnam,
*          lv_rel_pkg_name     TYPE klschl,
*          lv_version          TYPE atwrt,
*          lv_serviceable_flag TYPE c.
*    DATA: lv_error_flag TYPE c,
*          lv_error_msg  TYPE char256.
    DATA: lt_activity_groups TYPE STANDARD TABLE OF bapiagr,
          lt_return          TYPE TABLE OF bapiret2,
          lv_valid_action    TYPE abap_bool VALUE abap_false.
**Create Release Package declaration
    DATA: lv_bom_no     TYPE stko_api02-bom_no,
          lv_matnr      TYPE csap_mbom-matnr,
          lv_plant      TYPE csap_mbom-werks,
*          lv_bom_usage  TYPE csap_mbom-stlan,
          lv_valid_from TYPE csap_mbom-datuv.
    DATA: ls_return_1 TYPE bapireturn1,
          lt_mat_num  TYPE TABLE OF bapimatinr.
    DATA: i_stko TYPE stko_api01.
    DATA: ls_headdata            TYPE bapimathead,
          ls_clientdata          TYPE bapi_mara,
          ls_clientdatax         TYPE bapi_marax,
          ls_plantdata           TYPE bapi_marc,
          ls_plantdatax          TYPE bapi_marcx,
          ls_salesdata           TYPE bapi_mvke,
          ls_salesdatax          TYPE bapi_mvkex,
          ls_return_2            TYPE bapiret2,
          lt_materialdescription TYPE  TABLE OF bapi_makt,
          ls_materialdescription TYPE bapi_makt,
          lt_returnmessages      TYPE TABLE OF bapi_matreturn2,
          ls_returnmessages      TYPE bapi_matreturn2.
```

```
**Bapi Structure
*    DATA:   lt_return               TYPE TABLE OF bapiret2,
*            ls_return               TYPE bapiret2,
*            lt_charactdetailnew     TYPE TABLE OF bapicharactdetail,
*            ls_charactdetail        TYPE  bapicharactdetail,
*            lt_charactdescrnew      TYPE TABLE OF bapicharactdescr,
*            ls_charactdescrnew      TYPE  bapicharactdescr,
*            lt_charactvaluescharnew TYPE TABLE OF bapicharactvalueschar,
*            ls_charactvaluescharnew TYPE  bapicharactvalueschar,
*            lt_charactvaluesnumnew  TYPE TABLE OF bapicharactvaluesnum,
*            ls_charactvaluesnumnew  TYPE  bapicharactvaluesnum,
*            lt_charactvaluescurr    TYPE TABLE OF bapicharactvaluescurr,
*            ls_charactvaluescurr    TYPE  bapicharactvaluescurr,
*            lt_charactvaluesdescrnew TYPE TABLE OF bapicharactvaluesdescr,
*            ls_charactvaluesdescrnew TYPE  bapicharactvaluesdescr,
*            lt_charactreferences    TYPE TABLE OF bapicharactreferences,
*            ls_charactreferences    TYPE  bapicharactreferences,
*            lt_charactrestrictions  TYPE TABLE OF bapicharactrestrictions,
*            ls_charactrestrictions  TYPE  bapicharactrestrictions.
*
*    DATA: lt_bapi1003        TYPE TABLE OF bapi1003_alloc_values_num,
*          lt_values_char     TYPE TABLE OF bapi1003_alloc_values_char,
*          lt_values_char_new TYPE TABLE OF bapi1003_alloc_values_char,
*          lt_values_curr     TYPE TABLE OF bapi1003_alloc_values_curr,
*          lt_bapiret2        TYPE TABLE OF bapiret2,
*          ls_bapiret2        TYPE bapiret2,
*          ls_bapi1003        TYPE bapi1003_alloc_values_num,
*          ls_values_char     TYPE bapi1003_alloc_values_char,
*          ls_values_curr     TYPE bapi1003_alloc_values_char.
*
*    DATA: lv_classnum   TYPE klasse_d,
*          lv_class_type TYPE klassenart.
**Field Symbol Declaration
    FIELD-SYMBOLS <ls_parameter>      TYPE /iwbep/s_mgw_name_value_pair.
**Constant declaration
*    CONSTANTS: con_agr_name   TYPE agr_name VALUE 'ZS_AS_VERSION_UPDATE_MD'. "Adding
new roles - 07/27/2020
    CONSTANTS: con_role_endusr TYPE agr_name VALUE 'ZC_AS_PRODUCT_OWNERS_MD_ENDUSR',
"With out Tcode access
               con_role_md     TYPE agr_name VALUE 'ZC_AS_PRODUCT_OWNERS_MD'.
"With Tcode access well
**Retrieve key/value of incoming request parameters:
    lt_key_tab = io_tech_request_context->get_parameters( ).
*Retrieve user role data
    CALL FUNCTION 'BAPI_USER_GET_DETAIL'
      EXPORTING
        username       = sy-uname
      TABLES
        activitygroups = lt_activity_groups
        return         = lt_return.
*Check for the update role - If any of the MD role is assigned , then allow the user
to perform the action.
*else throw the error message
    CLEAR: lv_valid_action.
```

```
    READ TABLE lt_activity_groups WITH KEY agr_name = con_role_endusr TRANSPORTING NO
FIELDS.
    IF sy-subrc EQ 0.
      lv_valid_action = abap_true.
    ELSE.
      READ TABLE lt_activity_groups WITH KEY agr_name = con_role_md TRANSPORTING NO
FIELDS.
      IF sy-subrc EQ 0.
        lv_valid_action = abap_true.
      ENDIF.
    ENDIF.
    IF lv_valid_action EQ abap_true.
**For Version Master
      CLEAR: ls_key_tab,
             lv_ga_date.
      READ TABLE  lt_key_tab INTO ls_key_tab WITH KEY name = 'ZAS_GA_DATE'.
      IF sy-subrc = 0.
        lv_ga_date = ls_key_tab-value+0(8).
        IF lv_ga_date = '19000101'.
          lv_ga_date = '00000000'.
        ENDIF.
      ENDIF.
      CLEAR: ls_key_tab,
             lv_n_rank.
      READ TABLE  lt_key_tab INTO ls_key_tab WITH KEY name = 'ZAS_N_RANK'.
      IF sy-subrc = 0.
        lv_n_rank = ls_key_tab-value.
      ENDIF.
      CLEAR: ls_key_tab,
             lv_rp_desc.
      READ TABLE  lt_key_tab INTO ls_key_tab WITH KEY name = 'ZAS_RP_DESC'.
      IF sy-subrc = 0.
        lv_rp_desc = ls_key_tab-value.
      ENDIF.
      CLEAR: ls_key_tab,
             lv_rp_version.
      READ TABLE  lt_key_tab INTO ls_key_tab WITH KEY name = 'ZAS_RP_VERSION'.
      IF sy-subrc = 0.
        lv_rp_version = ls_key_tab-value.
      ENDIF.
      CLEAR: ls_key_tab,
             lv_serv_flag.
      READ TABLE  lt_key_tab INTO ls_key_tab WITH KEY name = 'ZAS_SERV_FLAG'.
      IF sy-subrc = 0.
        lv_serv_flag = ls_key_tab-value.
      ENDIF.
**For Material Master Release Package
      CLEAR: ls_key_tab,
             lv_product.
      READ TABLE  lt_key_tab INTO ls_key_tab WITH KEY name = 'PRODUCT'.
      IF sy-subrc = 0.
        lv_product = ls_key_tab-value.
      ENDIF.
      CLEAR: ls_key_tab,
```

```
              lv_rpdesc.
        READ TABLE  lt_key_tab INTO ls_key_tab WITH KEY name = 'RPDESC'.
        IF sy-subrc = 0.
          lv_rpdesc = ls_key_tab-value.
        ENDIF.
        CLEAR: ls_key_tab,
               lv_rpdesc_to.
        READ TABLE  lt_key_tab INTO ls_key_tab WITH KEY name = 'RPDESCT'.
        IF sy-subrc = 0.
          lv_rpdesc_to = ls_key_tab-value.
        ENDIF.
        CLEAR: ls_key_tab,
               lv_bsegment,
               lv_plant.
        READ TABLE  lt_key_tab INTO ls_key_tab WITH KEY name = 'BSEGMENT'.
        IF sy-subrc = 0.
          lv_bsegment = ls_key_tab-value.
          IF lv_bsegment EQ 'AS-Default'.
            lv_plant = '1001'.
          ELSEIF lv_bsegment EQ 'Airpas'.
            lv_plant = '3055'.
*Added Radixx  24/09/2021 Sg0311766
          ELSEIF lv_bsegment EQ 'Radixx'.
            lv_plant = '1024'.
          ENDIF.
        ENDIF.
**Get Release Package
*       READ TABLE lt_key_tab INTO DATA(ls_key_tab) WITH KEY name = 'REL_PKG_NAME'.
*       IF sy-subrc = 0 AND ls_key_tab-value IS NOT INITIAL.
*         CLEAR: lv_rel_pkg_name , lv_rel_pkg.
*         lv_rel_pkg_name = ls_key_tab-value.
**        lv_rel_pkg = ls_key_tab-value.
*         SELECT SINGLE internal_num
*                       char_name
*                       FROM zas_clas_rel_pkg
*                       INTO ( lv_rel_pkg_atinn,
*                              lv_rel_pkg )
*                       WHERE  class_desc EQ lv_rel_pkg_name.
*         IF sy-subrc = 0.
*         ENDIF.
*       ENDIF.
*
***Get Version
*       CLEAR:ls_key_tab.
*       READ TABLE  lt_key_tab INTO ls_key_tab WITH KEY name = 'VERSIONS'.
*       IF sy-subrc = 0 AND ls_key_tab-value IS NOT INITIAL.
*         CLEAR: lv_version.
*         lv_version = ls_key_tab-value.
*       ENDIF.
*
***Get serviceable flag
*       CLEAR:ls_key_tab.
*       READ TABLE  lt_key_tab INTO ls_key_tab WITH KEY name = 'SERVICEABLE'.
*       IF sy-subrc = 0 AND ls_key_tab-value IS NOT INITIAL.
```

```
*        CLEAR: lv_serviceable_flag.
*        lv_serviceable_flag = ls_key_tab-value.
*      ENDIF.
      CASE iv_action_name.
**Add Version In Between N Ranking
        WHEN 'AddVersionsBetween'.
**Fetch version details from version master
          SELECT zas_release_pkg,
                 zas_rp_version,
                 zas_rp_version_int,
                 zas_rp_desc,
                 zas_serv_flag,
                 zas_n_rank_int,
                 zas_ga_date,
                 created_by,
                 created_on,
                 created_at,
                 changed_by,
                 changed_on,
                 changed_at
                 FROM zas_rp_vrsn_mast
                 INTO TABLE @DATA(lt_rp_vrsn_mast)
                 WHERE zas_rp_desc EQ @lv_rp_desc.
          IF sy-subrc = 0.
            SORT lt_rp_vrsn_mast BY zas_n_rank_int DESCENDING.
            READ TABLE lt_rp_vrsn_mast INTO DATA(lwa_rp_vrsn_mast) WITH KEY
zas_rp_version = lv_rp_version.
            IF sy-subrc <> 0.
              READ TABLE lt_rp_vrsn_mast INTO lwa_rp_vrsn_mast INDEX 1.
              IF sy-subrc = 0.
                CLEAR: ls_rp_vrsn_mast.
                ls_rp_vrsn_mast-zas_release_pkg    =
lwa_rp_vrsn_mast-zas_release_pkg.
                ls_rp_vrsn_mast-zas_rp_version     = lv_rp_version.
**Get internal number from Number Range Object
                CLEAR: lv_number.
                CALL FUNCTION 'NUMBER_GET_NEXT'
                  EXPORTING
                    nr_range_nr             = con_nr_range
                    object                  = con_nr_object
                  IMPORTING
                    number                  = lv_number
                  EXCEPTIONS
                    interval_not_found      = 1
                    number_range_not_intern = 2
                    object_not_found        = 3
                    quantity_is_0           = 4
                    quantity_is_not_1       = 5
                    interval_overflow       = 6
                    buffer_overflow         = 7
                    OTHERS                  = 8.
                IF sy-subrc = 0.
                  ls_rp_vrsn_mast-zas_rp_version_int = lv_number.
                ENDIF.
```

```
              ls_rp_vrsn_mast-zas_rp_desc        = lwa_rp_vrsn_mast-zas_rp_desc.
              ls_rp_vrsn_mast-zas_serv_flag      = lv_serv_flag.
              IF lv_n_rank = 'N-1'.
                ls_rp_vrsn_mast-zas_n_rank_int = 1.
              ELSEIF lv_n_rank = 'N-2'.
                ls_rp_vrsn_mast-zas_n_rank_int = 2.
              ELSEIF lv_n_rank = '>N-2'.
                ls_rp_vrsn_mast-zas_n_rank_int = lwa_rp_vrsn_mast-zas_n_rank_int +
1.
              ENDIF.
              ls_rp_vrsn_mast-zas_ga_date        = lv_ga_date.
              ls_rp_vrsn_mast-created_by = sy-uname.
              ls_rp_vrsn_mast-created_on = sy-datum.
              ls_rp_vrsn_mast-created_at = sy-uzeit.
**Insert the New Version to Version Master Table
              INSERT zas_rp_vrsn_mast FROM ls_rp_vrsn_mast.
              IF sy-subrc = 0.
                CLEAR:ls_status.
                ls_status-identifier  = 1.
                ls_status-success     = abap_true.
                ls_status-message     = 'Version Added Successfully'.
              ENDIF.
              CLEAR: ls_rp_vrsn_mast,
                     lwa_rp_vrsn_mast.
            ENDIF.
          ELSE.
            CLEAR:ls_status.
            ls_status-identifier  = 1.
            ls_status-success     = abap_false.
            ls_status-message     = 'Version Already Exist. Please check it'.
          ENDIF.
        ELSE.
**If there is no entry in the Release Package Version Master table, then its a new
Release Package.
**Just to make sure check the Release Package exist in view ZASV_RPG_DATA
          SELECT SINGLE matnr_rpg,
                        rpg_desc,
                        werks,
                        stlan,
                        stlnr,
                        stlal,
                        datuv
                        FROM zasv_rpg_data
                        INTO @DATA(ls_rpg_det)
                        WHERE rpg_desc EQ @lv_rp_desc.
          IF sy-subrc = 0.
**Dont allow user to create the '>N-2' version directly , with out proceeding N
Ranked Version
            IF lv_n_rank <> '>N-2'.
              CLEAR: ls_rp_vrsn_mast.
              ls_rp_vrsn_mast-zas_release_pkg    = ls_rpg_det-matnr_rpg.
              ls_rp_vrsn_mast-zas_rp_version     = lv_rp_version.
**Get internal number from Number Range Object
              CLEAR: lv_number.
```

```
                CALL FUNCTION 'NUMBER_GET_NEXT'
                  EXPORTING
                    nr_range_nr             = con_nr_range
                    object                  = con_nr_object
                  IMPORTING
                    number                  = lv_number
                  EXCEPTIONS
                    interval_not_found      = 1
                    number_range_not_intern = 2
                    object_not_found        = 3
                    quantity_is_0           = 4
                    quantity_is_not_1       = 5
                    interval_overflow       = 6
                    buffer_overflow         = 7
                    OTHERS                  = 8.
                IF sy-subrc = 0.
                  ls_rp_vrsn_mast-zas_rp_version_int = lv_number.
                ENDIF.
                ls_rp_vrsn_mast-zas_rp_desc        = ls_rpg_det-rpg_desc.
                ls_rp_vrsn_mast-zas_serv_flag      = lv_serv_flag.
                CLEAR: ls_rp_vrsn_mast-zas_n_rank_int.
                IF lv_n_rank = 'N-1'.
                  ls_rp_vrsn_mast-zas_n_rank_int = 1.
                ELSEIF lv_n_rank = 'N-2'.
                  ls_rp_vrsn_mast-zas_n_rank_int = 2.
                ENDIF.
                ls_rp_vrsn_mast-zas_ga_date        = lv_ga_date.
                ls_rp_vrsn_mast-created_by         = sy-uname.
                ls_rp_vrsn_mast-created_on         = sy-datum.
                ls_rp_vrsn_mast-created_at         = sy-uzeit.
**Insert the New Version to Version Master Table
                INSERT zas_rp_vrsn_mast FROM ls_rp_vrsn_mast.
                IF sy-subrc = 0.
                  CLEAR:ls_status.
                  ls_status-identifier  = 1.
                  ls_status-success     = abap_true.
                  ls_status-message     = 'Version Added Successfully'.
                ENDIF.
                CLEAR: ls_rp_vrsn_mast,
                       ls_rpg_det.
**If user try to create '>N-2' rank with out prior N ranked version , then return
error message
               ELSE.
                  CLEAR:ls_status.
                  ls_status-identifier  = 1.
                  ls_status-success     = abap_false.
                  ls_status-message     = 'Please Maintain First  N Ranked Version'.
               ENDIF.
*              ELSE.
*                CLEAR:ls_status.
*                ls_status-identifier  = 1.
*                ls_status-success     = abap_false.
*                ls_status-message     = 'Release Package Not Exist in Version Master.
Please check it'.
```

```
            ENDIF.
          ENDIF.
**Add Version In Existing N Ranking
        WHEN 'AddVersionsExisting'.
**Fetch version details from version master
          REFRESH: lt_rp_vrsn_mast.
          SELECT zas_release_pkg,
                 zas_rp_version,
                 zas_rp_version_int,
                 zas_rp_desc,
                 zas_serv_flag,
                 zas_n_rank_int,
                 zas_ga_date,
                 created_by,
                 created_on,
                 created_at,
                 changed_by,
                 changed_on,
                 changed_at
                 FROM zas_rp_vrsn_mast
                 INTO TABLE @lt_rp_vrsn_mast
                 WHERE zas_rp_desc EQ @lv_rp_desc.
          IF sy-subrc = 0.
**Check if the version is already exist. If not then proceed further to create the
new version. Else return error message
            CLEAR: lwa_rp_vrsn_mast.
            READ TABLE lt_rp_vrsn_mast INTO lwa_rp_vrsn_mast WITH KEY zas_rp_version
= lv_rp_version.
            IF sy-subrc <> 0.
              READ TABLE lt_rp_vrsn_mast INTO lwa_rp_vrsn_mast INDEX 1.
              IF sy-subrc = 0.
                CLEAR: ls_rp_vrsn_mast.
                ls_rp_vrsn_mast-zas_release_pkg    =
lwa_rp_vrsn_mast-zas_release_pkg.
                ls_rp_vrsn_mast-zas_rp_version     = lv_rp_version.
**Get internal number from Number Range Object
                CLEAR: lv_number.
                CALL FUNCTION 'NUMBER_GET_NEXT'
                  EXPORTING
                    nr_range_nr             = con_nr_range
                    object                  = con_nr_object
                  IMPORTING
                    number                  = lv_number
                  EXCEPTIONS
                    interval_not_found      = 1
                    number_range_not_intern = 2
                    object_not_found        = 3
                    quantity_is_0           = 4
                    quantity_is_not_1       = 5
                    interval_overflow       = 6
                    buffer_overflow         = 7
                    OTHERS                  = 8.
                IF sy-subrc = 0.
                  ls_rp_vrsn_mast-zas_rp_version_int = lv_number.
```

```
              ENDIF.
              ls_rp_vrsn_mast-zas_rp_desc        = lwa_rp_vrsn_mast-zas_rp_desc.
              ls_rp_vrsn_mast-zas_serv_flag      = lv_serv_flag.
              ls_rp_vrsn_mast-zas_n_rank_int     = 0.
              ls_rp_vrsn_mast-zas_ga_date        = lv_ga_date.
              ls_rp_vrsn_mast-created_by         = sy-uname.
              ls_rp_vrsn_mast-created_on         = sy-datum.
              ls_rp_vrsn_mast-created_at         = sy-uzeit.
**Insert the New Version to Version Master Table
              INSERT zas_rp_vrsn_mast FROM ls_rp_vrsn_mast.
              IF sy-subrc = 0.
                CLEAR:ls_status.
                ls_status-identifier  = 1.
                ls_status-success     = abap_true.
                ls_status-message     = 'Version Added Successfully'.
              ENDIF.
              CLEAR: ls_rp_vrsn_mast,
                     lwa_rp_vrsn_mast.
            ENDIF.
          ELSE.
            CLEAR:ls_status.
            ls_status-identifier  = 1.
            ls_status-success     = abap_false.
            ls_status-message     = 'Version Already Exist. Please check it'.
          ENDIF.
        ELSE.
**If there is no entry in the Release Package Version Master table, then its a new
Release Package.
**Just to make sure check the Release Package exist in view ZASV_RPG_DATA
          CLEAR: ls_rpg_det.
          SELECT SINGLE matnr_rpg,
                        rpg_desc,
                        werks,
                        stlan,
                        stlnr,
                        stlal,
                        datuv
                        FROM zasv_rpg_data
                        INTO @ls_rpg_det
                        WHERE rpg_desc EQ @lv_rp_desc.
          IF sy-subrc = 0.
            CLEAR: ls_rp_vrsn_mast.
            ls_rp_vrsn_mast-zas_release_pkg    = ls_rpg_det-matnr_rpg.
            ls_rp_vrsn_mast-zas_rp_version     = lv_rp_version.
**Get internal number from Number Range Object
            CLEAR: lv_number.
            CALL FUNCTION 'NUMBER_GET_NEXT'
              EXPORTING
                nr_range_nr             = con_nr_range
                object                  = con_nr_object
              IMPORTING
                number                  = lv_number
              EXCEPTIONS
                interval_not_found      = 1
```

```
                  number_range_not_intern = 2
                  object_not_found        = 3
                  quantity_is_0           = 4
                  quantity_is_not_1       = 5
                  interval_overflow       = 6
                  buffer_overflow         = 7
                  OTHERS                  = 8.
              IF sy-subrc = 0.
                ls_rp_vrsn_mast-zas_rp_version_int = lv_number.
              ENDIF.
              ls_rp_vrsn_mast-zas_rp_desc        = ls_rpg_det-rpg_desc.
              ls_rp_vrsn_mast-zas_serv_flag      = lv_serv_flag.
              ls_rp_vrsn_mast-zas_n_rank_int     = 0.
              ls_rp_vrsn_mast-zas_ga_date        = lv_ga_date.
              ls_rp_vrsn_mast-created_by         = sy-uname.
              ls_rp_vrsn_mast-created_on         = sy-datum.
              ls_rp_vrsn_mast-created_at         = sy-uzeit.
**Insert the New Version to Version Master Table
              INSERT zas_rp_vrsn_mast FROM ls_rp_vrsn_mast.
              IF sy-subrc = 0.
                CLEAR:ls_status.
                ls_status-identifier  = 1.
                ls_status-success     = abap_true.
                ls_status-message     = 'Version Added Successfully'.
              ENDIF.
              CLEAR: ls_rp_vrsn_mast,
                     ls_rpg_det.
*            ELSE.
*              CLEAR:ls_status.
*              ls_status-identifier  = 1.
*              ls_status-success     = abap_false.
*              ls_status-message     = 'Release Package Not Exist in Version Master.
Please check it'.
            ENDIF.
          ENDIF.
**Add Brand New N Version
        WHEN 'AddVersionsNew'.
**Fetch version details from version master
          REFRESH: lt_rp_vrsn_mast,
                   lt_rp_vrsn_mast_upd.
          SELECT zas_release_pkg,
                 zas_rp_version,
                 zas_rp_version_int,
                 zas_rp_desc,
                 zas_serv_flag,
                 zas_n_rank_int,
                 zas_ga_date,
                 created_by,
                 created_on,
                 created_at,
                 changed_by,
                 changed_on,
                 changed_at
                 FROM zas_rp_vrsn_mast
```

```
                INTO TABLE @lt_rp_vrsn_mast
                WHERE zas_rp_desc EQ @lv_rp_desc.
          IF sy-subrc = 0.
            SORT lt_rp_vrsn_mast BY zas_n_rank_int.
*Check if the version is already exist. If not then proceed further to create the new
version. Else return error message
            READ TABLE lt_rp_vrsn_mast INTO lwa_rp_vrsn_mast WITH KEY zas_rp_version
= lv_rp_version.
            IF sy-subrc <> 0.
              CLEAR: lwa_rp_vrsn_mast,
                     lv_count.
              lv_count = 0.
              LOOP AT lt_rp_vrsn_mast INTO lwa_rp_vrsn_mast.
                lv_count = lv_count + 1.
                IF lv_count = 1.
                  CLEAR: ls_rp_vrsn_mast.
                  ls_rp_vrsn_mast-zas_release_pkg    =
lwa_rp_vrsn_mast-zas_release_pkg.
                  ls_rp_vrsn_mast-zas_rp_version     = lv_rp_version.
**Get internal number from Number Range Object
                  CLEAR: lv_number.
                  CALL FUNCTION 'NUMBER_GET_NEXT'
                    EXPORTING
                      nr_range_nr             = con_nr_range
                      object                  = con_nr_object
                    IMPORTING
                      number                  = lv_number
                    EXCEPTIONS
                      interval_not_found      = 1
                      number_range_not_intern = 2
                      object_not_found        = 3
                      quantity_is_0           = 4
                      quantity_is_not_1       = 5
                      interval_overflow       = 6
                      buffer_overflow         = 7
                      OTHERS                  = 8.
                  IF sy-subrc = 0.
                    ls_rp_vrsn_mast-zas_rp_version_int = lv_number.
                  ENDIF.
                  ls_rp_vrsn_mast-zas_rp_desc        = lwa_rp_vrsn_mast-zas_rp_desc.
                  ls_rp_vrsn_mast-zas_serv_flag      = lv_serv_flag.
                  ls_rp_vrsn_mast-zas_n_rank_int     = 0.
                  ls_rp_vrsn_mast-zas_ga_date        = lv_ga_date.
                  ls_rp_vrsn_mast-created_by         = sy-uname.
                  ls_rp_vrsn_mast-created_on         = sy-datum.
                  ls_rp_vrsn_mast-created_at         = sy-uzeit.
                  APPEND ls_rp_vrsn_mast TO lt_rp_vrsn_mast_upd.
                  CLEAR: ls_rp_vrsn_mast.
                ENDIF.
                ls_rp_vrsn_mast-zas_release_pkg     =
lwa_rp_vrsn_mast-zas_release_pkg.
                ls_rp_vrsn_mast-zas_rp_version      =
lwa_rp_vrsn_mast-zas_rp_version.
                ls_rp_vrsn_mast-zas_rp_version_int  =
lwa_rp_vrsn_mast-zas_rp_version_int.
```

```
              ls_rp_vrsn_mast-zas_rp_desc         = lwa_rp_vrsn_mast-zas_rp_desc.
              ls_rp_vrsn_mast-zas_serv_flag       = lwa_rp_vrsn_mast-zas_serv_flag.
              ls_rp_vrsn_mast-zas_n_rank_int      = lwa_rp_vrsn_mast-zas_n_rank_int
+ 01.
              ls_rp_vrsn_mast-zas_ga_date         = lwa_rp_vrsn_mast-zas_ga_date.
              ls_rp_vrsn_mast-created_by          = lwa_rp_vrsn_mast-created_by.
              ls_rp_vrsn_mast-created_on          = lwa_rp_vrsn_mast-created_on.
              ls_rp_vrsn_mast-created_at          = lwa_rp_vrsn_mast-created_at.
              ls_rp_vrsn_mast-changed_by          = sy-uname.
              ls_rp_vrsn_mast-changed_on          = sy-datum.
              ls_rp_vrsn_mast-changed_at          = sy-uzeit.
              APPEND ls_rp_vrsn_mast TO lt_rp_vrsn_mast_upd.
              CLEAR: ls_rp_vrsn_mast,
                     lwa_rp_vrsn_mast.
            ENDLOOP.
            IF lt_rp_vrsn_mast_upd[] IS NOT INITIAL.
              MODIFY zas_rp_vrsn_mast FROM TABLE lt_rp_vrsn_mast_upd.
              IF sy-subrc = 0.
                CLEAR:ls_status.
                ls_status-identifier  = 1.
                ls_status-success     = abap_true.
                ls_status-message     = 'Version Added Successfully'.
              ENDIF.
              REFRESH: lt_rp_vrsn_mast_upd.
            ENDIF.
          ELSE.
            CLEAR:ls_status.
            ls_status-identifier  = 1.
            ls_status-success     = abap_false.
            ls_status-message     = 'Version Already Exist. Please check it'.
          ENDIF.
        ELSE.
**If there is no entry in the Release Package Version Master table, then its a new
Release Package.
**Just to make sure check the Release Package exist in view ZASV_RPG_DATA
          CLEAR: ls_rpg_det.
          SELECT SINGLE matnr_rpg,
                        rpg_desc,
                        werks,
                        stlan,
                        stlnr,
                        stlal,
                        datuv
                        FROM zasv_rpg_data
                        INTO @ls_rpg_det
                        WHERE rpg_desc EQ @lv_rp_desc.
          IF sy-subrc = 0.
            CLEAR: ls_rp_vrsn_mast.
            ls_rp_vrsn_mast-zas_release_pkg    = ls_rpg_det-matnr_rpg.
            ls_rp_vrsn_mast-zas_rp_version     = lv_rp_version.
**Get internal number from Number Range Object
            CLEAR: lv_number.
            CALL FUNCTION 'NUMBER_GET_NEXT'
              EXPORTING
```

```
                nr_range_nr             = con_nr_range
                object                  = con_nr_object
              IMPORTING
                number                  = lv_number
              EXCEPTIONS
                interval_not_found      = 1
                number_range_not_intern = 2
                object_not_found        = 3
                quantity_is_0           = 4
                quantity_is_not_1       = 5
                interval_overflow       = 6
                buffer_overflow         = 7
                OTHERS                  = 8.
            IF sy-subrc = 0.
              ls_rp_vrsn_mast-zas_rp_version_int = lv_number.
            ENDIF.
            ls_rp_vrsn_mast-zas_rp_desc        = ls_rpg_det-rpg_desc.
            ls_rp_vrsn_mast-zas_serv_flag      = lv_serv_flag.
            ls_rp_vrsn_mast-zas_n_rank_int     = 0.
            ls_rp_vrsn_mast-zas_ga_date        = lv_ga_date.
            ls_rp_vrsn_mast-created_by         = sy-uname.
            ls_rp_vrsn_mast-created_on         = sy-datum.
            ls_rp_vrsn_mast-created_at         = sy-uzeit.
**Insert the New Version to Version Master Table
            INSERT zas_rp_vrsn_mast FROM ls_rp_vrsn_mast.
            IF sy-subrc = 0.
              CLEAR:ls_status.
              ls_status-identifier  = 1.
              ls_status-success     = abap_true.
              ls_status-message     = 'Version Added Successfully'.
            ENDIF.
            CLEAR: ls_rp_vrsn_mast,
                   ls_rpg_det.
*           ELSE.
*             CLEAR:ls_status.
*             ls_status-identifier  = 1.
*             ls_status-success     = abap_false.
*             ls_status-message     = 'Release Package Not Exist in Version Master.
Please check it'.
          ENDIF.
        ENDIF.
**Create New Release Package
       WHEN 'CreateReleasePkg'.
**This contains couple of activity.First Material Creation and then BOM Creation
against the material.
**Check if the Release package is already exist or not. If not then create the
Release Package
        SELECT SINGLE rpg_desc
                      FROM zasv_rpg_data
                      INTO @DATA(ls_rpg_data)
                      WHERE rpg_desc EQ @lv_rpdesc.
        IF sy-subrc <> 0.
          CLEAR: ls_return_1.
          REFRESH: lt_mat_num.
```

```
**Get Next available number for material to create
            CALL FUNCTION 'BAPI_STDMATERIAL_GETINTNUMBER'
              EXPORTING
                material_type    = con_matype
                industry_sector  = con_indsr
                required_numbers = 1
              IMPORTING
                return           = ls_return_1
              TABLES
                material_number  = lt_mat_num.
**Get the Material number
            CLEAR: ls_headdata.
            READ TABLE lt_mat_num INTO DATA(ls_mat_num) INDEX 1.
            IF sy-subrc = 0.
              ls_headdata-material = ls_mat_num-material.
            ENDIF.
**Move header details to bapi work area
            ls_headdata-ind_sector = con_indsr.
            ls_headdata-matl_type =  con_matype.
            ls_headdata-basic_view = abap_true.
            ls_headdata-sales_view = abap_true.
**Move client details to bapi work area
            CLEAR: ls_clientdata.
            ls_clientdata-base_uom = con_uom_ea.
            ls_clientdata-item_cat = con_item_cat.
**Move client checkbox details to bapi work area
            CLEAR: ls_clientdatax.
            ls_clientdatax-base_uom = abap_true.
            ls_clientdatax-item_cat = abap_true.
**Move plant details to bapi work area
            CLEAR: ls_plantdata-plant.
            ls_plantdata-plant = lv_plant.
**Move plant check box to bapi work area
            CLEAR: ls_plantdatax.
            ls_plantdatax-plant = lv_plant.
**Move Material Description to Bapi work area
            REFRESH: lt_materialdescription.
            CLEAR: ls_materialdescription.
            ls_materialdescription-langu = con_lang.
            ls_materialdescription-matl_desc = lv_rpdesc.
            APPEND ls_materialdescription TO lt_materialdescription.
            CLEAR: ls_materialdescription.
**Call BAPI to create the Material
            CLEAR: ls_return_2.
            REFRESH: lt_returnmessages.
            CALL FUNCTION 'BAPI_MATERIAL_SAVEDATA'
              EXPORTING
                headdata            = ls_headdata
                clientdata          = ls_clientdata
                clientdatax         = ls_clientdatax
                plantdata           = ls_plantdata
                plantdatax          = ls_plantdatax
              IMPORTING
                return              = ls_return_2
```

```
              TABLES
                materialdescription = lt_materialdescription
                returnmessages      = lt_returnmessages.
             READ TABLE lt_returnmessages INTO ls_returnmessages WITH KEY type = 'E'.
             IF sy-subrc <> 0.
**If no error , then commit to save the changes
              CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
                EXPORTING
                  wait = 'X'.
**Process to create the BOM
              CLEAR: lv_matnr,
                     lv_valid_from,
                     lv_bom_no,
                     lv_warning,
                     i_stko.
**Material
              lv_matnr = ls_mat_num-material.
**Convert the date to external format
              CALL FUNCTION 'CONVERT_DATE_TO_EXTERNAL'
                EXPORTING
                  date_internal            = sy-datum
                IMPORTING
                  date_external            = lv_valid_from
                EXCEPTIONS
                  date_internal_is_invalid = 1
                  OTHERS                   = 2.
**Call FM to create BOM
              CALL FUNCTION 'CSAP_MAT_BOM_CREATE'
                EXPORTING
                  material           = lv_matnr
                  plant              = lv_plant
                  bom_usage          = con_bom_usage
                  valid_from         = lv_valid_from
                  i_stko             = i_stko
                  fl_commit_and_wait = 'X'
                IMPORTING
                  fl_warning         = lv_warning
                  bom_no             = lv_bom_no
                EXCEPTIONS
                  error              = 1
                  OTHERS             = 2.
              IF sy-subrc = 0 AND  lv_bom_no IS NOT INITIAL.
**BOM is Created
                CLEAR:ls_status.
                ls_status-identifier  = 1.
                ls_status-success     = abap_true.
                ls_status-message     = 'Release Package Created Successfully'.
              ELSE.
                CLEAR:ls_status.
                ls_status-identifier  = 1.
                ls_status-success     = abap_false.
                ls_status-message     = 'Error ocuured while creating the BOM. Please
check it.'.
              ENDIF.
```

```
            ELSE.
              CLEAR:ls_status.
              ls_status-identifier  = 1.
              ls_status-success     = abap_false.
              ls_status-message     = 'Error ocuured while creating the Material.
Please check it.'.
            ENDIF.
          ELSE.
            CLEAR:ls_status.
            ls_status-identifier  = 1.
            ls_status-success     = abap_false.
            ls_status-message     = 'Release Package with Same Name already exist.
Please check it'.
          ENDIF.
          CLEAR: ls_rpg_data.
**Remove Product From The Release Package Assignement - BOM
        WHEN 'RemoveProduct'.
**Get the BOM Header details from CDS view ZASV_RPG_DATA
          SELECT SINGLE matnr_rpg,
                        rpg_desc,
                        werks,
                        stlan,
                        datuv
                        FROM zasv_rpg_data
                        INTO @DATA(ls_header_rpg_data)
                        WHERE rpg_desc = @lv_rpdesc.
          IF sy-subrc = 0.
            CLEAR: lv_date.
            CALL FUNCTION 'CONVERT_DATE_TO_EXTERNAL'
              EXPORTING
                date_internal            = ls_header_rpg_data-datuv
              IMPORTING
                date_external            = lv_date
              EXCEPTIONS
                date_internal_is_invalid = 1
                OTHERS                   = 2.
            CALL FUNCTION 'CALO_INIT_API'
              EXCEPTIONS
                log_object_not_found     = 1
                log_sub_object_not_found = 2
                OTHERS                   = 3.
            CLEAR: ls_stk2,
                   lv_warning,
                   lt_stp2[].
            DO 5 TIMES.
              CALL FUNCTION 'CSAP_MAT_BOM_OPEN'
                EXPORTING
                  material   = ls_header_rpg_data-matnr_rpg
                  plant      = ls_header_rpg_data-werks
                  bom_usage  = ls_header_rpg_data-stlan
                  valid_from = lv_date
                IMPORTING
                  o_stko     = ls_stk2
                  fl_warning = lv_warning
```

```
    TABLES
      t_stpo     = lt_stp2
    EXCEPTIONS
      error      = 1
      OTHERS     = 2.
  IF sy-subrc = 0.
    EXIT.
  ENDIF.
  WAIT UP TO 2 SECONDS.
ENDDO.
IF sy-subrc = 0.
  CLEAR: ls_stp2.
  READ TABLE lt_stp2 INTO ls_stp2  WITH KEY component = lv_product.
  IF sy-subrc = 0.
    ls_stp2-fldelete = abap_true.
    CLEAR: lv_warning,
               lt_dep_source[].
    CALL FUNCTION 'CSAP_BOM_ITEM_MAINTAIN'
      EXPORTING
        i_stpo       = ls_stp2
      IMPORTING
        fl_warning   = lv_warning
      TABLES
        t_dep_source = lt_dep_source
      EXCEPTIONS
        error        = 1
        OTHERS       = 2.
    CLEAR: lv_warning.
    CALL FUNCTION 'CSAP_MAT_BOM_CLOSE'
      IMPORTING
        fl_warning = lv_warning
      EXCEPTIONS
        error      = 1
        OTHERS     = 2.
    IF sy-subrc = 0  AND lv_warning IS INITIAL.
      CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
        EXPORTING
          wait = abap_true.
      CLEAR:ls_status.
      ls_status-identifier  = 1.
      ls_status-success     = abap_true.
      ls_status-message     = 'Product Removed Successfully'.
    ELSE.
      CLEAR:ls_status.
      ls_status-identifier  = 1.
      ls_status-success     = abap_false.
      ls_status-message     = 'Error'.
    ENDIF.
  ENDIF.
ELSE.
  CLEAR:ls_status.
  ls_status-identifier  = 1.
  ls_status-success     = abap_false.
  ls_status-message     = 'BOM Locked'.
```

```
          ENDIF.
        ENDIF.
**Insert New Product to the Release Package Assignement - BOM
       WHEN 'InsertProduct'.
**Get the BOM Header details from CDS view ZASV_RPG_DATA
        CLEAR: ls_header_rpg_data.
        SELECT SINGLE matnr_rpg,
                      rpg_desc,
                      werks,
                      stlan,
                      datuv
                      FROM zasv_rpg_data
                      INTO @ls_header_rpg_data
                      WHERE rpg_desc = @lv_rpdesc.
        IF sy-subrc = 0.
          CLEAR: lv_date.
          CALL FUNCTION 'CONVERT_DATE_TO_EXTERNAL'
            EXPORTING
              date_internal            = ls_header_rpg_data-datuv
            IMPORTING
              date_external            = lv_date
            EXCEPTIONS
              date_internal_is_invalid = 1
              OTHERS                   = 2.
          CALL FUNCTION 'CALO_INIT_API'
            EXCEPTIONS
              log_object_not_found     = 1
              log_sub_object_not_found = 2
              OTHERS                   = 3.
          CLEAR: ls_stk2,
                 lv_warning,
                 lt_stp2[].
          CALL FUNCTION 'CSAP_MAT_BOM_OPEN'
            EXPORTING
              material   = ls_header_rpg_data-matnr_rpg
              plant      = ls_header_rpg_data-werks
              bom_usage  = ls_header_rpg_data-stlan
              valid_from = lv_date
            IMPORTING
              o_stko     = ls_stk2
              fl_warning = lv_warning
            TABLES
              t_stpo     = lt_stp2
            EXCEPTIONS
              error      = 1
              OTHERS     = 2.
          IF sy-subrc = 0.
            CLEAR: ls_stp2.
            ls_stp2-item_categ = con_item_categ.
            ls_stp2-component = lv_product.
            ls_stp2-comp_qty = con_comp_qty.
            CLEAR: lv_warning,
                        lt_dep_source[].
            CALL FUNCTION 'CSAP_BOM_ITEM_MAINTAIN'
```

```
              EXPORTING
                i_stpo       = ls_stp2
              IMPORTING
                fl_warning   = lv_warning
              TABLES
                t_dep_source = lt_dep_source
              EXCEPTIONS
                error        = 1
                OTHERS       = 2.
            CLEAR: lv_warning.
            CALL FUNCTION 'CSAP_MAT_BOM_CLOSE'
              IMPORTING
                fl_warning = lv_warning
              EXCEPTIONS
                error      = 1
                OTHERS     = 2.
            IF sy-subrc = 0 AND lv_warning IS INITIAL.
              CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
                EXPORTING
                  wait = abap_true.
              CLEAR:ls_status.
              ls_status-identifier  = 1.
              ls_status-success     = abap_true.
              ls_status-message     = 'Product assigned to a Release Package
Successfully'.
            ELSE.
              CLEAR:ls_status.
              ls_status-identifier  = 1.
              ls_status-success     = abap_false.
              ls_status-message     = 'Error'.
            ENDIF.
          ELSE.
            CLEAR:ls_status.
            ls_status-identifier  = 1.
            ls_status-success     = abap_false.
            ls_status-message     = 'BOM Locked'.
          ENDIF.
        ENDIF.
**Reassign Product From one Release Package to Another Release Package Assignement -
BOM
      WHEN 'ReassignProduct'.
**First we need to delete the Product from the Existing Rlease Package Assignement.
Then Insert into new Release Package.
**Get the BOM Header details from CDS view ZASV_RPG_DATA
        SELECT matnr_rpg,
               rpg_desc,
               werks,
               stlan,
               datuv
               FROM zasv_rpg_data
               INTO TABLE @DATA(lt_header_rpg_data)
               WHERE rpg_desc IN ( @lv_rpdesc , @lv_rpdesc_to ).
        IF sy-subrc = 0.
          CLEAR: ls_header_rpg_data.
```

```
            READ TABLE lt_header_rpg_data INTO ls_header_rpg_data WITH KEY rpg_desc =
lv_rpdesc.
            IF sy-subrc = 0.
              CLEAR: lv_date.
              CALL FUNCTION 'CONVERT_DATE_TO_EXTERNAL'
                EXPORTING
                  date_internal            = ls_header_rpg_data-datuv
                IMPORTING
                  date_external            = lv_date
                EXCEPTIONS
                  date_internal_is_invalid = 1
                  OTHERS                   = 2.
              CALL FUNCTION 'CALO_INIT_API'
                EXCEPTIONS
                  log_object_not_found     = 1
                  log_sub_object_not_found = 2
                  OTHERS                   = 3.
              CLEAR: ls_stk2,
                   lv_warning,
                   lt_stp2[].
              CALL FUNCTION 'CSAP_MAT_BOM_OPEN'
                EXPORTING
                  material   = ls_header_rpg_data-matnr_rpg
                  plant      = ls_header_rpg_data-werks
                  bom_usage  = ls_header_rpg_data-stlan
                  valid_from = lv_date
                IMPORTING
                  o_stko     = ls_stk2
                  fl_warning = lv_warning
                TABLES
                  t_stpo     = lt_stp2
                EXCEPTIONS
                  error      = 1
                  OTHERS     = 2.
              IF sy-subrc = 0.
                CLEAR: ls_stp2.
                READ TABLE lt_stp2 INTO ls_stp2  WITH KEY component = lv_product.
                IF sy-subrc = 0.
                  ls_stp2-fldelete = abap_true.
                  CLEAR: lv_warning,
                           lt_dep_source[].
                  CALL FUNCTION 'CSAP_BOM_ITEM_MAINTAIN'
                    EXPORTING
                      i_stpo       = ls_stp2
                    IMPORTING
                      fl_warning   = lv_warning
                    TABLES
                      t_dep_source = lt_dep_source
                    EXCEPTIONS
                      error        = 1
                      OTHERS       = 2.
                  CLEAR: lv_warning.
                  CALL FUNCTION 'CSAP_MAT_BOM_CLOSE'
                    IMPORTING
```

```
                    fl_warning = lv_warning
                  EXCEPTIONS
                    error      = 1
                    OTHERS     = 2.
                CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
                  EXPORTING
                    wait = abap_true.
              ENDIF.
**Insert to the new release Package
**Get the BOM Header details from CDS view ZASV_RPG_DATA
              CLEAR: ls_header_rpg_data.
              READ TABLE lt_header_rpg_data INTO ls_header_rpg_data WITH KEY
rpg_desc = lv_rpdesc_to.
              IF sy-subrc = 0.
                CLEAR: lv_date.
                CALL FUNCTION 'CONVERT_DATE_TO_EXTERNAL'
                  EXPORTING
                    date_internal            = ls_header_rpg_data-datuv
                  IMPORTING
                    date_external            = lv_date
                  EXCEPTIONS
                    date_internal_is_invalid = 1
                    OTHERS                   = 2.
                CALL FUNCTION 'CALO_INIT_API'
                  EXCEPTIONS
                    log_object_not_found     = 1
                    log_sub_object_not_found = 2
                    OTHERS                   = 3.
                CLEAR: ls_stk2,
                 lv_warning,
                 lt_stp2[].
                CALL FUNCTION 'CSAP_MAT_BOM_OPEN'
                  EXPORTING
                    material   = ls_header_rpg_data-matnr_rpg
                    plant      = ls_header_rpg_data-werks
                    bom_usage  = ls_header_rpg_data-stlan
                    valid_from = lv_date
                  IMPORTING
                    o_stko     = ls_stk2
                    fl_warning = lv_warning
                  TABLES
                    t_stpo     = lt_stp2
                  EXCEPTIONS
                    error      = 1
                    OTHERS     = 2.
                IF sy-subrc = 0.
                  CLEAR: ls_stp2.
                  ls_stp2-item_categ = con_item_categ.
                  ls_stp2-component = lv_product.
                  ls_stp2-comp_qty = con_comp_qty.
                  CLEAR: lv_warning,
                         lt_dep_source[].
                  CALL FUNCTION 'CSAP_BOM_ITEM_MAINTAIN'
                    EXPORTING
```

```
                      i_stpo       = ls_stp2
                    IMPORTING
                      fl_warning   = lv_warning
                    TABLES
                      t_dep_source = lt_dep_source
                    EXCEPTIONS
                      error        = 1
                      OTHERS       = 2.
                  CLEAR: lv_warning.
                  CALL FUNCTION 'CSAP_MAT_BOM_CLOSE'
                    IMPORTING
                      fl_warning = lv_warning
                    EXCEPTIONS
                      error      = 1
                      OTHERS     = 2.
                  IF sy-subrc = 0 AND lv_warning IS INITIAL.
                    CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
                      EXPORTING
                        wait = abap_true.
                    CLEAR:ls_status.
                    ls_status-identifier  = 1.
                    ls_status-success     = abap_true.
                    ls_status-message     = 'Product Reassigned to a Release
Package Successfully'.
                  ELSE.
                    ls_status-identifier  = 1.
                    ls_status-success     = abap_false.
                    ls_status-message     = 'Failed to Reassign the Product'.
                  ENDIF.
                ELSE.
                  ls_status-identifier  = 1.
                  ls_status-success     = abap_false.
                  ls_status-message     = 'BOM is Locked'.
                ENDIF.
              ENDIF.
            ENDIF.
          ENDIF.
        ENDIF.
      WHEN 'SetServiceableFlag'.
        CLEAR: lwa_rp_vrsn_mast.
        SELECT SINGLE zas_release_pkg,
                      zas_rp_version,
                      zas_rp_version_int,
                      zas_rp_desc,
                      zas_serv_flag,
                      zas_n_rank_int,
                      zas_ga_date,
                      created_by,
                      created_on,
                      created_at,
                      changed_by,
                      changed_on,
                      changed_at
                      FROM zas_rp_vrsn_mast
```

```
                        INTO @lwa_rp_vrsn_mast
                        WHERE zas_rp_desc = @lv_rp_desc
                        AND   zas_rp_version = @lv_rp_version.
          IF sy-subrc = 0.
            CLEAR: ls_rp_vrsn_mast.
            ls_rp_vrsn_mast-zas_release_pkg    = lwa_rp_vrsn_mast-zas_release_pkg.
            ls_rp_vrsn_mast-zas_rp_version     = lwa_rp_vrsn_mast-zas_rp_version.
            ls_rp_vrsn_mast-zas_rp_version_int = lwa_rp_vrsn_mast-zas_rp_version_int.
            ls_rp_vrsn_mast-zas_rp_desc        = lwa_rp_vrsn_mast-zas_rp_desc.
            ls_rp_vrsn_mast-zas_serv_flag      = lv_serv_flag .
            ls_rp_vrsn_mast-zas_n_rank_int     = lwa_rp_vrsn_mast-zas_n_rank_int.
            ls_rp_vrsn_mast-zas_ga_date        = lwa_rp_vrsn_mast-zas_ga_date.
            ls_rp_vrsn_mast-created_by          = lwa_rp_vrsn_mast-created_by.
            ls_rp_vrsn_mast-created_on          = lwa_rp_vrsn_mast-created_on.
            ls_rp_vrsn_mast-created_at          = lwa_rp_vrsn_mast-created_at.
            ls_rp_vrsn_mast-changed_by          = sy-uname.
            ls_rp_vrsn_mast-changed_on          = sy-datum.
            ls_rp_vrsn_mast-changed_at          = sy-uzeit.
**Set the Serviceable flag with Updated Change log
            MODIFY zas_rp_vrsn_mast FROM ls_rp_vrsn_mast.
            IF sy-subrc = 0.
              CLEAR:ls_status.
              ls_status-identifier  = 1.
              ls_status-success     = abap_true.
              ls_status-message     = 'Success'.
            ELSE.
              CLEAR:ls_status.
              ls_status-identifier  = 1.
              ls_status-success     = abap_true.
              ls_status-message     = 'Update Failed'.
            ENDIF.
            CLEAR: ls_rp_vrsn_mast.
          ENDIF.
*         WHEN 'AddVersions'.
*           IF lv_version IS NOT INITIAL AND
*              lv_rel_pkg IS NOT INITIAL.
*            REFRESH: lt_return.
***Get the Character detail
*             CALL FUNCTION 'BAPI_CHARACT_GETDETAIL'
*               EXPORTING
*                 charactname        = lv_rel_pkg
*               IMPORTING
*                 charactdetail      = ls_charactdetail
*               TABLES
*                 charactdescr       = lt_charactdescrnew
*                 charactvalueschar  = lt_charactvaluescharnew
*                 charactvaluesdescr = lt_charactvaluesdescrnew
*                 return             = lt_return.
***Check the Version is already exist or not?? If exist throw the message
*             READ TABLE lt_charactvaluescharnew INTO DATA(ls_charval) WITH KEY
value_char = lv_version.
*             IF sy-subrc NE 0.
***Fill the BAPI structure with the new details
*               APPEND ls_charactdetail TO lt_charactdetailnew.
```

```
*              CLEAR: ls_charactdetail.
***Move char value
*              ls_charactvaluescharnew-value_char = lv_version.
*              APPEND ls_charactvaluescharnew TO lt_charactvaluescharnew.
*              CLEAR: ls_charactvaluescharnew.
***Move char Value Description
*              ls_charactvaluesdescrnew-language_int = sy-langu.
*              WRITE  ls_charactvaluesdescrnew-language_int  TO
ls_charactvaluesdescrnew-language_iso.
*              ls_charactvaluesdescrnew-value_char =  lv_version.
*              ls_charactvaluesdescrnew-description = lv_version.
*              APPEND ls_charactvaluesdescrnew TO lt_charactvaluesdescrnew.
*              CLEAR: ls_charactvaluesdescrnew.
*              REFRESH: lt_return.
***Add the new version to characteristic
*              CALL FUNCTION 'BAPI_CHARACT_CHANGE'
*                EXPORTING
*                  charactname           = lv_rel_pkg
*                TABLES
*                  charactdetailnew      = lt_charactdetailnew
*                  charactdescrnew       = lt_charactdescrnew
*                  charactvaluescharnew  = lt_charactvaluescharnew
*                  charactvaluesdescrnew = lt_charactvaluesdescrnew
*                  return                = lt_return.
*              READ TABLE lt_return INTO ls_return WITH KEY type = 'E'.
*              IF sy-subrc NE 0.
*                CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
*                  EXPORTING
*                    wait = 'X'.
*                CLEAR: ls_status.
*                ls_status-identifier  = 1.
*                ls_status-success     = abap_true.
*                ls_status-message     = 'Master Version Updated Successfully'.
***Check the versionable flag. If its X , then we need to assign the new version to
all the materials.
*                IF lv_serviceable_flag EQ 'Y'.
*                  SELECT object_number,
*                                    class_type,
*                                    class,
*                                    class_desc
*                                    FROM zas_matmas_ver
*                                    INTO TABLE @DATA(lt_matmas_ver)
*                                    WHERE class_desc EQ @lv_rel_pkg_name.
*                  IF sy-subrc = 0.
*                  ENDIF.
*                  LOOP AT lt_matmas_ver INTO DATA(ls_matmas_ver).
***Pass the Char Value to the Vc structure
*                    CLEAR: ls_values_char. REFRESH:lt_values_char_new.
*                    ls_values_char-charact    = lv_rel_pkg.
"Characteristics Name
*                    ls_values_char-value_char = lv_version .  "characteristics value
*                    APPEND ls_values_char TO lt_values_char_new.
*                    CLEAR: lt_values_char.
*                    REFRESH: lt_bapi1003[] , lt_values_char[], lt_values_curr[],
lt_bapiret2[].
```

```
***Get the existing vc values first.
*                     CALL FUNCTION 'BAPI_OBJCL_GETDETAIL'
*                       EXPORTING
*                         objectkey       = ls_matmas_ver-object_number
*                         objecttable     = 'MARA'
*                         classnum        = ls_matmas_ver-class
*                         classtype       = ls_matmas_ver-class_type
*                       TABLES
*                         allocvaluesnum  = lt_bapi1003
*                         allocvalueschar = lt_values_char
*                         allocvaluescurr = lt_values_curr
*                         return          = lt_bapiret2.
***Check its already exist or not??? If not update it
*                     CLEAR: ls_values_char.
*                     READ TABLE lt_values_char INTO ls_values_char WITH KEY charact
= lv_rel_pkg
*
value_char = lv_version.
*                     IF sy-subrc NE 0.
*                       IF lt_values_char_new[] IS NOT INITIAL.
*                         APPEND LINES OF lt_values_char_new TO lt_values_char.
*                         REFRESH: lt_values_char_new[].
*                       ENDIF.
*                     ELSE.
*                       REFRESH: lt_values_char_new.
*                       CONTINUE.
*                     ENDIF.
***Call bapi to update the VC value
*                     CALL FUNCTION 'BAPI_OBJCL_CHANGE'
*                       EXPORTING
*                         objectkey          = ls_matmas_ver-object_number
*                         objecttable        = 'MARA'
*                         classnum           = ls_matmas_ver-class
*                         classtype          = ls_matmas_ver-class_type
*                       TABLES
*                         allocvaluesnumnew  = lt_bapi1003
*                         allocvaluescharnew = lt_values_char
*                         allocvaluescurrnew = lt_values_curr
*                         return             = lt_bapiret2.
*                     IF sy-subrc = 0.
*                       CLEAR: ls_bapiret2.
*                       READ TABLE lt_bapiret2 INTO ls_bapiret2 WITH KEY type = 'E'.
*                       IF sy-subrc NE 0.
*                         CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
*                           EXPORTING
*                             wait = 'X'.
*                       ELSE.
*                         CLEAR:lv_error_flag,
*                               lv_error_msg.
*                         lv_error_flag = abap_true.
*                         CLEAR: ls_status.
*                         ls_status-identifier  = 3.
*                         ls_status-success     = abap_false.
*                         CLEAR: ls_bapiret2.
```

```
*                       LOOP AT lt_bapiret2 INTO ls_bapiret2 WHERE type = 'E'.
*                         CONCATENATE ls_status-message ls_bapiret2-message INTO
ls_status-message SEPARATED BY space.
*                       ENDLOOP.
*                     ENDIF.
*                   ENDIF.
*                 ENDLOOP.
*               ENDIF.
*             ELSE.
*               CALL FUNCTION 'BAPI_TRANSACTION_ROLLBACK'.
*               CLEAR: ls_status.
*               ls_status-identifier  = 3.
*               ls_status-success     = abap_false.
*
*               LOOP AT lt_return INTO ls_return WHERE type = 'E'.
*                 CONCATENATE ls_status-message ls_return-message INTO
ls_status-message SEPARATED BY space.
*               ENDLOOP.
*             ENDIF.
*           ELSE.
*             CLEAR: ls_status.
*             ls_status-identifier  = 3.
*             ls_status-success     = abap_false.
*             ls_status-message     = 'Entry with Release Package Name and Version
already exists.'.
*           ENDIF.
*         ELSE.
*           CLEAR: ls_status.
*           ls_status-identifier  = 3.
*           ls_status-success     = abap_false.
*           ls_status-message     = 'Please check the input values'.
*         ENDIF.
*       WHEN 'SetServiceableFlag'.
*         IF lv_version IS NOT INITIAL AND
*           lv_rel_pkg IS NOT INITIAL.
***Get all the products which is assigned to a release package
*           SELECT object_number
*                  class_type
*                  class
*                  class_desc
*                  FROM zas_matmas_ver
*                  INTO TABLE lt_matmas_ver
*                  WHERE class_desc EQ lv_rel_pkg_name.
*           IF sy-subrc = 0.
*           ENDIF.
*           IF lv_serviceable_flag EQ 'Y'.
*             LOOP AT lt_matmas_ver INTO ls_matmas_ver.
***Pass the Char Value to the Vc structure
*               CLEAR: ls_values_char. REFRESH: lt_values_char_new.
*               ls_values_char-charact    = lv_rel_pkg.
"Characteristics Name
*               ls_values_char-value_char = lv_version .  "characteristics value
*               APPEND ls_values_char TO lt_values_char_new.
*               CLEAR: ls_values_char.
```

```
*                REFRESH: lt_bapi1003[] , lt_values_char[], lt_values_curr[],
lt_bapiret2[].
***Get the existing vc values first.
*                CALL FUNCTION 'BAPI_OBJCL_GETDETAIL'
*                  EXPORTING
*                    objectkey       = ls_matmas_ver-object_number
*                    objecttable     = 'MARA'
*                    classnum        = ls_matmas_ver-class
*                    classtype       = ls_matmas_ver-class_type
*                  TABLES
*                    allocvaluesnum  = lt_bapi1003
*                    allocvalueschar = lt_values_char
*                    allocvaluescurr = lt_values_curr
*                    return          = lt_bapiret2.
***Check its already exist or not??? If not update it
*                CLEAR: ls_values_char.
*                READ TABLE lt_values_char INTO ls_values_char WITH KEY charact    =
lv_rel_pkg
*                                                                        value_char =
lv_version.
*                IF sy-subrc NE 0.
*                  IF lt_values_char_new[] IS NOT INITIAL.
*                    APPEND LINES OF lt_values_char_new TO lt_values_char.
*                    REFRESH: lt_values_char_new[].
*                  ENDIF.
*                ELSE.
*                  REFRESH: lt_values_char_new.
*                  CONTINUE.
*                ENDIF.
***Call bapi to update the VC value
*                CALL FUNCTION 'BAPI_OBJCL_CHANGE'
*                  EXPORTING
*                    objectkey          = ls_matmas_ver-object_number
*                    objecttable        = 'MARA'
*                    classnum           = ls_matmas_ver-class
*                    classtype          = ls_matmas_ver-class_type
*                  TABLES
*                    allocvaluesnumnew  = lt_bapi1003
*                    allocvaluescharnew = lt_values_char
*                    allocvaluescurrnew = lt_values_curr
*                    return             = lt_bapiret2.
*                IF sy-subrc = 0.
*                  CLEAR: ls_bapiret2.
*                  READ TABLE lt_bapiret2 INTO ls_bapiret2 WITH KEY type = 'E'.
*                  IF sy-subrc NE 0.
*                    CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
*                      EXPORTING
*                        wait = 'X'.
*                  ELSE.
*                    CALL FUNCTION 'BAPI_TRANSACTION_ROLLBACK'.
*                    CLEAR:lv_error_flag,
*                              lv_error_msg.
*                    lv_error_flag = abap_true.
*                    CLEAR: ls_status.
```

```
*                    ls_status-identifier  = 3.
*                    ls_status-success     = abap_false.
*                    CLEAR: ls_bapiret2.
*                    LOOP AT lt_bapiret2 INTO ls_bapiret2 WHERE type = 'E'.
*                      CONCATENATE ls_status-message ls_bapiret2-message INTO
ls_status-message SEPARATED BY space.
*                    ENDLOOP.
*
*                  ENDIF.
*                ENDIF.
*              ENDLOOP.
*            ELSE.
***Remove the serviceable flag
*              LOOP AT lt_matmas_ver INTO ls_matmas_ver.
*                REFRESH: lt_bapi1003[] , lt_values_char[], lt_values_curr[],
lt_bapiret2[].
***Get the existing vc values first.
*                CALL FUNCTION 'BAPI_OBJCL_GETDETAIL'
*                  EXPORTING
*                    objectkey       = ls_matmas_ver-object_number
*                    objecttable     = 'MARA'
*                    classnum        = ls_matmas_ver-class
*                    classtype       = ls_matmas_ver-class_type
*                  TABLES
*                    allocvaluesnum  = lt_bapi1003
*                    allocvalueschar = lt_values_char
*                    allocvaluescurr = lt_values_curr
*                    return          = lt_bapiret2.
*
*                DELETE lt_values_char WHERE  value_char =  lv_version .
***Call bapi to update the VC value
*                CALL FUNCTION 'BAPI_OBJCL_CHANGE'
*                  EXPORTING
*                    objectkey          = ls_matmas_ver-object_number
*                    objecttable        = 'MARA'
*                    classnum           = ls_matmas_ver-class
*                    classtype          = ls_matmas_ver-class_type
*                  TABLES
*                    allocvaluesnumnew  = lt_bapi1003
*                    allocvaluescharnew = lt_values_char
*                    allocvaluescurrnew = lt_values_curr
*                    return             = lt_bapiret2.
*                IF sy-subrc = 0.
*                  CLEAR: ls_bapiret2.
*                  READ TABLE lt_bapiret2 INTO ls_bapiret2 WITH KEY type = 'E'.
*                  IF sy-subrc NE 0.
*                    CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
*                      EXPORTING
*                        wait = 'X'.
*                  ELSE.
*                    CALL FUNCTION 'BAPI_TRANSACTION_ROLLBACK'.
*                    CLEAR:lv_error_flag,
*                              lv_error_msg.
*                    lv_error_flag = abap_true.
```

```
*                    CLEAR: ls_status.
*                    ls_status-identifier  = 3.
*                    ls_status-success     = abap_false.
*                    CLEAR: ls_bapiret2.
*                    LOOP AT lt_bapiret2 INTO ls_bapiret2 WHERE type = 'E'.
*                      CONCATENATE ls_status-message ls_bapiret2-message INTO
ls_status-message SEPARATED BY space.
*                    ENDLOOP.
*                  ENDIF.
*                ENDIF.
*              ENDLOOP.
*            ENDIF.
*            IF lv_error_flag IS INITIAL.
*              CLEAR: ls_status.
*              ls_status-identifier  = 1.
*              ls_status-success     = abap_true.
*              ls_status-message     = 'Master Version Updated Successfully'.
*            ENDIF.
*          ELSE.
*            CLEAR: ls_status.
*            ls_status-identifier  = 3.
*            ls_status-success     = abap_false.
*            ls_status-message     = 'Update Failed'.
*          ENDIF.
      ENDCASE.
    ELSE.
**Auth issue
      ls_status-identifier  = 2.              "Access Issue - No Auth
      ls_status-success     = abap_false.
      ls_status-message     = 'You are not authorized. Please contact security team'.
    ENDIF.
    me->copy_data_to_ref(
 EXPORTING
   is_data = ls_status
 CHANGING
   cr_data = er_data ).
  ENDMETHOD.
```

IF_SADL_GW_QUERY_CONTROL~SET_QUERY_OPTIONS

```
  METHOD if_sadl_gw_query_control~set_query_options.
**TRY.
*CALL METHOD SUPER->IF_SADL_GW_QUERY_CONTROL~SET_QUERY_OPTIONS
*  EXPORTING
*    IV_ENTITY_SET    =
*    IO_QUERY_OPTIONS =
*    .
** CATCH /iwbep/cx_mgw_busi_exception .
** CATCH /iwbep/cx_mgw_tech_exception .
**ENDTRY.
    DATA lt_parameters    TYPE if_sadl_public_types=>tt_entity_parameters.
    DATA lt_sort_elements TYPE if_sadl_public_types=>tt_sort_elements.
    DATA lt_search_scope  TYPE if_sadl_public_types=>tt_search_scope.
    DATA lt_req_elements  TYPE if_sadl_public_types=>tt_requested_elements.
```

```
    DATA lo_provider      TYPE REF TO if_sadl_cond_prov_auth_objects.
    DATA lv_search_term   TYPE string.
    CASE iv_entity_set.
      WHEN 'AssignedProductSHSet'.
        "Set search scope to include all visible/requested (above) fields:
        lt_search_scope = VALUE #( ( `MATNR` ) ( `MAKTX` ) ).
        io_query_options->set_text_search_scope( lt_search_scope ).
        lv_search_term = io_query_options->get_text_search_term( ).
        lv_search_term = `*` && lv_search_term && `*`.
        io_query_options->set_text_search_term( lv_search_term ).
      WHEN 'UnAssignedProductSHSet'.
        "Set search scope to include all visible/requested (above) fields:
        lt_search_scope = VALUE #( ( `MATNR` ) ( `MAKTX` ) ).
        io_query_options->set_text_search_scope( lt_search_scope ).
        lv_search_term = io_query_options->get_text_search_term( ).
        lv_search_term = `*` && lv_search_term && `*`.
        io_query_options->set_text_search_term( lv_search_term ).
      WHEN OTHERS.
    ENDCASE.
  ENDMETHOD.
```

MATMASRELPKGDATA_GET_ENTITYSET

```
  METHOD matmasrelpkgdata_get_entityset.
**TRY.
*CALL METHOD SUPER->MATMASRELPKGDATA_GET_ENTITYSET
*  EXPORTING
*    IV_ENTITY_NAME           =
*    IV_ENTITY_SET_NAME       =
*    IV_SOURCE_NAME           =
*    IT_FILTER_SELECT_OPTIONS =
*    IS_PAGING                =
*    IT_KEY_TAB               =
*    IT_NAVIGATION_PATH       =
*    IT_ORDER                 =
*    IV_FILTER_STRING         =
*    IV_SEARCH_STRING         =
**    io_tech_request_context  =
**  IMPORTING
**    et_entityset             =
**    es_response_context      =
*    .
** CATCH /iwbep/cx_mgw_busi_exception .
** CATCH /iwbep/cx_mgw_tech_exception .
**ENDTRY.
**Structure Declaration
    TYPES: BEGIN OF st_final_tab,
             material             TYPE matnr,
             material_description TYPE maktx,
**<< Begin of insert by vijay on 05/14/2020 - DS0K956809 Enable Export All Function
             class                TYPE klasse_d,
**>> End of insert by vijay on 05/14/2020
             class_desc           TYPE klschl,
             prdha                TYPE prodh_d,
```

```
             prodh1               TYPE prodh_d,
             vtext1               TYPE bezei40,
             prodh2               TYPE prodh_d,
             vtext2               TYPE bezei40,
             prodh3               TYPE prodh_d,
             vtext3               TYPE bezei40,
             prodh4               TYPE prodh_d,
             vtext4               TYPE bezei40,
           END OF st_final_tab.
**Internal table and Workarea declaration
    DATA: lt_final_tab TYPE TABLE OF st_final_tab,
          ls_final_tab TYPE          st_final_tab.
    DATA: ls_entity LIKE LINE OF et_entityset.
    DATA: lo_filter TYPE  REF TO /iwbep/if_mgw_req_filter.
    DATA: lt_filter_select_options TYPE /iwbep/t_mgw_select_option.
    DATA: ls_filter TYPE /iwbep/s_mgw_select_option.
    DATA: lv_filter_str TYPE string.
    DATA: ls_converted_keys LIKE LINE OF et_entityset.
    DATA: ls_order     TYPE /iwbep/s_mgw_sorting_order.
    DATA: lv_skip TYPE int4,
          lv_top  TYPE int4.
**Structure and Internal table for input parameters
    DATA: lr_rel_pkg_name LIKE RANGE OF ls_converted_keys-rel_pkg_name,
          ls_rel_pkg_name LIKE LINE OF lr_rel_pkg_name.
* Get filter or select option information
    lo_filter = io_tech_request_context->get_filter( ).
    lt_filter_select_options = lo_filter->get_filter_select_options( ).
    lv_filter_str = lo_filter->get_filter_string( ).
    IF lt_filter_select_options IS NOT INITIAL.
      LOOP AT lt_filter_select_options INTO ls_filter.
        CASE ls_filter-property.
          WHEN 'REL_PKG_NAME'.
            lo_filter->convert_select_option(
              EXPORTING
                is_select_option = ls_filter
              IMPORTING
                et_select_option = lr_rel_pkg_name ).
          WHEN OTHERS.
        ENDCASE.
      ENDLOOP.
    ENDIF.
**    IF NOT lr_rel_pkg_name[] IS INITIAL.              "Comment by Vijay on
05/14/2020 - DS0K956809 Enable Export All Function
**Get Material Master Release Package data from CDS View ZAS_MATMAS_VER
    SELECT material
           material_description
**<< Begin of insert by vijay on 05/14/2020 - DS0K956809 Enable Export All Function
           class
**>> End of insert by vijay on 05/14/2020
           class_desc
           prdha
           FROM zas_matmas_ver
           INTO TABLE lt_final_tab
           WHERE class_desc  IN lr_rel_pkg_name.
```

```
    IF sy-subrc = 0.
      LOOP AT lt_final_tab ASSIGNING FIELD-SYMBOL(<fs_final>).
        IF NOT <fs_final>-prdha IS INITIAL.
          <fs_final>-prodh1 = <fs_final>-prdha+0(5).
          <fs_final>-prodh2 = <fs_final>-prdha+0(9).
          <fs_final>-prodh3 = <fs_final>-prdha+0(14).
          <fs_final>-prodh4 = <fs_final>-prdha+0(18).
        ENDIF.
      ENDLOOP.
    ENDIF.
    IF lt_final_tab[] IS NOT INITIAL.
**Get the product hierarhy description
      SELECT prodh,
             vtext
             INTO TABLE @DATA(lt_suite)
             FROM t179t
             FOR ALL ENTRIES IN @lt_final_tab
             WHERE prodh EQ @lt_final_tab-prodh1.
      IF sy-subrc = 0.
      ENDIF.
      SELECT prodh,
             vtext
             INTO TABLE @DATA(lt_sol_family)
             FROM t179t
             FOR ALL ENTRIES IN @lt_final_tab
             WHERE prodh EQ @lt_final_tab-prodh2.
      IF sy-subrc = 0.
      ENDIF.
      SELECT prodh,
             vtext
             INTO TABLE @DATA(lt_solution)
             FROM t179t
             FOR ALL ENTRIES IN @lt_final_tab
             WHERE prodh EQ @lt_final_tab-prodh3.
      IF sy-subrc = 0.
      ENDIF.
      SELECT prodh,
             vtext
             INTO TABLE @DATA(lt_capability)
             FROM t179t
             FOR ALL ENTRIES IN @lt_final_tab
             WHERE prodh EQ @lt_final_tab-prodh4.
      IF sy-subrc = 0.
      ENDIF.
*  determine skip and top parameters
*  ---------------------------------------------------------------------
*        IF is_paging-skip IS NOT INITIAL.
*          lv_skip = is_paging-skip + 1.
*        ENDIF.
*
*        IF is_paging-top <> 0
*        AND lv_skip IS NOT INITIAL.
*          lv_top = is_paging-top + lv_skip - 1.
*        ELSEIF is_paging-top <> 0
```

```
*        AND lv_skip IS INITIAL.
*          lv_top = is_paging-top.
*        ELSE.
*          lv_top = lines( lt_final_tab ).
*        ENDIF.
      REFRESH: et_entityset.
      SORT lt_final_tab BY class_desc material.
      UNASSIGN: <fs_final>.
      LOOP AT lt_final_tab ASSIGNING <fs_final>.     "FROM lv_skip TO lv_top.
**  Get Suite Description
        READ TABLE lt_suite INTO DATA(ls_suite) WITH KEY prodh = <fs_final>-prodh1.
        IF sy-subrc = 0.
          <fs_final>-vtext1 = ls_suite-vtext.
        ENDIF.
**  Get Solution Family Description
        READ TABLE lt_sol_family INTO DATA(ls_sol_family) WITH KEY prodh =
<fs_final>-prodh2.
        IF sy-subrc = 0.
          <fs_final>-vtext2 = ls_sol_family-vtext.
        ENDIF.
**  Get Solution Description
        READ TABLE lt_solution INTO DATA(ls_solution) WITH KEY prodh =
<fs_final>-prodh3.
        IF sy-subrc = 0.
          <fs_final>-vtext3 = ls_solution-vtext.
        ENDIF.
**  Get Capability Description
        READ TABLE lt_capability INTO DATA(ls_capability) WITH KEY prodh =
<fs_final>-prodh4.
        IF sy-subrc = 0.
          <fs_final>-vtext4 = ls_capability-vtext.
        ENDIF.
**  Move the values to entity set
        ls_entity-rel_pkg_name = <fs_final>-class_desc.
**<< Begin of insert by vijay on 05/14/2020 - DS0K956809 Enable Export All Function
        ls_entity-class        = <fs_final>-class.
**>> End of insert by vijay on 05/14/2020
        ls_entity-product = <fs_final>-material.
        ls_entity-material_descr = <fs_final>-material_description.
        ls_entity-suite = <fs_final>-vtext1.
        ls_entity-solution_family = <fs_final>-vtext2.
        ls_entity-solution = <fs_final>-vtext3.
        ls_entity-capability = <fs_final>-vtext4.
        APPEND  ls_entity TO et_entityset.
        CLEAR: ls_entity.
      ENDLOOP.
    ENDIF.
** $inlinecount query option for all count entries.
    IF io_tech_request_context->has_inlinecount( ) = abap_true.
      DESCRIBE TABLE et_entityset LINES es_response_context-inlinecount.
    ELSE.
      CLEAR: es_response_context-inlinecount.
    ENDIF.
*** The function module for $top and $skip Query Options
```

```
    CALL METHOD /iwbep/cl_mgw_data_util=>paging
      EXPORTING
        is_paging = is_paging
      CHANGING
        ct_data   = et_entityset.
***The function module for Orderby condition
*        CALL METHOD /iwbep/cl_mgw_data_util=>orderby
*          EXPORTING
*            it_order = it_order
*          CHANGING
*            ct_data  = et_entityset.
    READ TABLE it_order INTO ls_order INDEX 1.
    IF sy-subrc = 0.
      IF ls_order-order EQ 'asc'.
        CASE ls_order-property.
          WHEN 'RelPkgName'.
            SORT et_entityset BY rel_pkg_name ASCENDING.
**<< Begin of insert by vijay on 05/14/2020 - DS0K956809 Enable Export All Function
          WHEN 'Class'.
            SORT et_entityset BY class ASCENDING.
**>> End of insert by vijay on 05/14/2020
          WHEN 'Product'.
            SORT et_entityset BY product ASCENDING.
          WHEN 'MaterialDescr'.
            SORT et_entityset BY material_descr ASCENDING.
          WHEN 'Suite'.
            SORT et_entityset BY suite ASCENDING.
          WHEN 'SolutionFamily'.
            SORT et_entityset BY solution_family ASCENDING.
          WHEN 'Solution'.
            SORT et_entityset BY solution ASCENDING.
          WHEN 'Capability'.
            SORT et_entityset BY capability ASCENDING.
        ENDCASE.
      ELSEIF ls_order-order EQ 'desc'.
        CASE ls_order-property.
          WHEN 'RelPkgName'.
            SORT et_entityset BY rel_pkg_name DESCENDING.
**<< Begin of insert by vijay on 05/14/2020 - DS0K956809 Enable Export All Function
          WHEN 'Class'.
            SORT et_entityset BY class DESCENDING.
**>> End of insert by vijay on 05/14/2020
          WHEN 'Product'.
            SORT et_entityset BY product DESCENDING.
          WHEN 'MaterialDescr'.
            SORT et_entityset BY material_descr DESCENDING.
          WHEN 'Suite'.
            SORT et_entityset BY suite DESCENDING.
          WHEN 'SolutionFamily'.
            SORT et_entityset BY solution_family DESCENDING.
          WHEN 'Solution'.
            SORT et_entityset BY solution DESCENDING.
          WHEN 'Capability'.
            SORT et_entityset BY capability DESCENDING.
```

```
        ENDCASE.
      ENDIF.
    ENDIF.
*    ENDIF.
  ENDMETHOD.
```

RELEASEPKGVERSIO_GET_ENTITYSET

```
  METHOD releasepkgversio_get_entityset.
**TRY.
*CALL METHOD SUPER->RELEASEPKGVERSIO_GET_ENTITYSET
*  EXPORTING
*    IV_ENTITY_NAME           =
*    IV_ENTITY_SET_NAME       =
*    IV_SOURCE_NAME           =
*    IT_FILTER_SELECT_OPTIONS =
*    IS_PAGING                =
*    IT_KEY_TAB               =
*    IT_NAVIGATION_PATH       =
*    IT_ORDER                 =
*    IV_FILTER_STRING         =
*    IV_SEARCH_STRING         =
**    io_tech_request_context  =
**  IMPORTING
**    et_entityset             =
**    es_response_context      =
*    .
** CATCH /iwbep/cx_mgw_busi_exception .
** CATCH /iwbep/cx_mgw_tech_exception .
**ENDTRY.
**Internal table and Workarea declaration
    DATA: lt_final_tab TYPE TABLE OF zas_rel_pkg,
          ls_final_tab TYPE          zas_rel_pkg.
    DATA: ls_entity LIKE LINE OF et_entityset.
    DATA: lo_filter TYPE  REF TO /iwbep/if_mgw_req_filter.
    DATA: lt_filter_select_options TYPE /iwbep/t_mgw_select_option.
    DATA: ls_filter TYPE /iwbep/s_mgw_select_option.
    DATA: lv_filter_str TYPE string.
    DATA: ls_converted_keys LIKE LINE OF et_entityset.
    DATA: ls_order     TYPE /iwbep/s_mgw_sorting_order.
**Structure and Internal table for input parameters
    DATA: lr_rel_pkg_name LIKE RANGE OF ls_converted_keys-rel_pkg_name,
          ls_rel_pkg_name LIKE LINE OF lr_rel_pkg_name.
* Get filter or select option information
    lo_filter = io_tech_request_context->get_filter( ).
    lt_filter_select_options = lo_filter->get_filter_select_options( ).
    lv_filter_str = lo_filter->get_filter_string( ).
    IF lt_filter_select_options IS NOT INITIAL.
      LOOP AT lt_filter_select_options INTO ls_filter.
        CASE ls_filter-property.
          WHEN 'REL_PKG_NAME'.
            lo_filter->convert_select_option(
              EXPORTING
                is_select_option = ls_filter
```

```
              IMPORTING
                et_select_option = lr_rel_pkg_name ).
           WHEN OTHERS.
         ENDCASE.
       ENDLOOP.
     ENDIF.
*    IF NOT lr_rel_pkg_name[] IS INITIAL. 'Commented by Vijay on 05/14/2020 to enable
export all function
**Get Release Package and its version from CDS View ZAS_REL_PKG_VRSN
    SELECT release_pkg_name
           version
           FROM zas_rel_pkg_vrsn
           INTO TABLE lt_final_tab
           WHERE release_pkg_name IN lr_rel_pkg_name.
    IF sy-subrc = 0.
**Get one product for which Release package name is same as input release package
name
**<< Begin of comment by vijay on 05/14/2020
*        SELECT SINGLE object_number,
*                      class_desc
*                      FROM zas_mat_rel_pkg
*                      INTO @DATA(ls_mat_rel_pkg)
*                      WHERE  class_desc  IN @lr_rel_pkg_name.
**>> End of comment by vijay on 05/14/2020
**<< Begin of insert by vijay on 05/14/2020
**To get the class name(Technical name)
      SELECT class,
             class_desc
             FROM zas_clas_rel_pkg
             INTO TABLE @DATA(lt_clas_rel_pkg)
             FOR ALL ENTRIES IN @lt_final_tab
             WHERE  class_desc  EQ @lt_final_tab-rel_pkg_name.
      IF sy-subrc = 0.
      ENDIF.
**Get the all the product assigned to release package
      SELECT object_number,
             class,
             class_desc
             FROM zas_mat_rel_pkg
             INTO TABLE @DATA(lt_mat_rel_pkg)
             FOR ALL ENTRIES IN @lt_final_tab
             WHERE  class_desc  EQ @lt_final_tab-rel_pkg_name.
**>> End of insert by vijay on 05/14/2020
      IF sy-subrc = 0.
**Keep only one product for each release package, to validate versionable or not.
**Delete rest of the products
        SORT lt_mat_rel_pkg BY class object_number.
        DELETE ADJACENT DUPLICATES FROM lt_mat_rel_pkg COMPARING class.
**<< Begin of comment by vijay on 05/14/2020
*          SELECT object_number,
*                 class_desc,
*                 characteristic_value
*                 FROM zas_prod_rel_pkg
*                 INTO TABLE @DATA(lt_prod_rel_pkg)
```

```
*                 WHERE object_number EQ @ls_mat_rel_pkg-object_number
*                 AND    class_desc EQ  @ls_mat_rel_pkg-class_desc.
**>> End of comment by vijay on 05/14/2020
**<< Begin of insert by vijay on 05/14/2020
**Get the assigned version details for products
        SELECT object_number,
               class_desc,
               characteristic_value
               FROM zas_prod_rel_pkg
               INTO TABLE @DATA(lt_prod_rel_pkg)
               FOR ALL ENTRIES IN @lt_mat_rel_pkg
               WHERE object_number EQ @lt_mat_rel_pkg-object_number
               AND    class_desc EQ  @lt_mat_rel_pkg-class_desc.
        IF sy-subrc = 0.
        ENDIF.
**>> End of insert by vijay on 05/14/2020
        LOOP AT lt_final_tab ASSIGNING FIELD-SYMBOL(<fs_final>).
**Randomly pick one product and check all the versions are assigned to it. If its
assigned, then consider its as versionable.
**If its not assigned , then its not versionable
          READ TABLE lt_prod_rel_pkg WITH KEY class_desc = <fs_final>-rel_pkg_name
                                              characteristic_value =
<fs_final>-versions
                                              TRANSPORTING NO FIELDS.
          IF sy-subrc = 0.
            <fs_final>-serviceable = 'Y'.
          ELSE.
            <fs_final>-serviceable = 'N'.
          ENDIF.
**<< Begin of insert by vijay on 05/14/2020 - To get Class Name
          READ TABLE lt_clas_rel_pkg INTO DATA(ls_clas_rel_pkg) WITH KEY class_desc =
<fs_final>-rel_pkg_name.
          IF sy-subrc = 0.
            <fs_final>-class = ls_clas_rel_pkg-class.
          ENDIF.
**>> End of insert by vijay on 05/14/2020
        ENDLOOP.
      ENDIF.
    ENDIF.
    IF lt_final_tab[] IS NOT INITIAL.
      et_entityset[] = lt_final_tab[].
**<< Begin of insert by vijay on 05/14/2020
      SORT  et_entityset BY rel_pkg_name
                            versions.
**>> End of insert by vijay on 05/14/2020
    ENDIF.
** $inlinecount query option for all count entries.
    IF io_tech_request_context->has_inlinecount( ) = abap_true.
      DESCRIBE TABLE et_entityset LINES es_response_context-inlinecount.
    ELSE.
      CLEAR: es_response_context-inlinecount.
    ENDIF.
*** The function module for $top and $skip Query Options
    CALL METHOD /iwbep/cl_mgw_data_util=>paging
```

```
      EXPORTING
        is_paging = is_paging
      CHANGING
        ct_data   = et_entityset.
**To enable sort functionality
    READ TABLE it_order INTO ls_order INDEX 1.
    IF sy-subrc = 0.
      IF ls_order-order EQ 'asc'.
        CASE ls_order-property.
          WHEN 'RelPkgName'.
            SORT et_entityset BY rel_pkg_name ASCENDING.
          WHEN 'Versions'.
            SORT et_entityset BY versions ASCENDING.
          WHEN 'Serviceable'.
            SORT et_entityset BY serviceable ASCENDING.
          WHEN 'Class'.
            SORT et_entityset BY class ASCENDING.
        ENDCASE.
      ELSEIF ls_order-order EQ 'desc'.
        CASE ls_order-property.
          WHEN 'RelPkgName'.
            SORT et_entityset BY rel_pkg_name DESCENDING.
          WHEN 'Versions'.
            SORT et_entityset BY versions DESCENDING.
          WHEN 'Serviceable'.
            SORT et_entityset BY serviceable DESCENDING.
          WHEN 'Class'.
            SORT et_entityset BY class DESCENDING.
        ENDCASE.
      ENDIF.
    ENDIF.
*    ENDIF.
  ENDMETHOD.
```

RPMATERIALMASTER_GET_ENTITYSET

```
  METHOD rpmaterialmaster_get_entityset.
**TRY.
*CALL METHOD SUPER->RPMATERIALMASTER_GET_ENTITYSET
*  EXPORTING
*    IV_ENTITY_NAME           =
*    IV_ENTITY_SET_NAME       =
*    IV_SOURCE_NAME           =
*    IT_FILTER_SELECT_OPTIONS =
*    IS_PAGING                =
*    IT_KEY_TAB               =
*    IT_NAVIGATION_PATH       =
*    IT_ORDER                 =
*    IV_FILTER_STRING         =
*    IV_SEARCH_STRING         =
**    io_tech_request_context  =
**  IMPORTING
**    et_entityset             =
**    es_response_context      =
```

```
*    .
** CATCH /iwbep/cx_mgw_busi_exception .
** CATCH /iwbep/cx_mgw_tech_exception .
**ENDTRY.
**Structure Declaration
    TYPES: BEGIN OF st_final_tab,
             matnr_rpg TYPE matnr,
             rpg_desc  TYPE maktx,
             matnr     TYPE matnr,
             maktx     TYPE maktx,
             prdha     TYPE prodh_d,
             prodh1    TYPE prodh_d,
             vtext1    TYPE bezei40,
             prodh2    TYPE prodh_d,
             vtext2    TYPE bezei40,
             prodh3    TYPE prodh_d,
             vtext3    TYPE bezei40,
             prodh4    TYPE prodh_d,
             vtext4    TYPE bezei40,
           END OF st_final_tab.
**Internal table and Workarea declaration
    DATA: lt_final_tab TYPE TABLE OF st_final_tab,
          ls_final_tab TYPE          st_final_tab.
    DATA: ls_entity LIKE LINE OF et_entityset.
    DATA: lo_filter TYPE  REF TO /iwbep/if_mgw_req_filter.
    DATA: lt_filter_select_options TYPE /iwbep/t_mgw_select_option.
    DATA: ls_filter TYPE /iwbep/s_mgw_select_option.
    DATA: lv_filter_str TYPE string.
    DATA: ls_converted_keys LIKE LINE OF et_entityset.
    DATA: lv_skip TYPE int4,
          lv_top  TYPE int4.
    DATA: lt_order TYPE /iwbep/t_mgw_tech_order.
    DATA: ls_order     TYPE /iwbep/s_mgw_tech_order.
    DATA: lt_otab  TYPE abap_sortorder_tab,
          ls_oline TYPE abap_sortorder.
**Structure and Internal table for input parameters
    DATA: lr_rpdesc  LIKE RANGE OF ls_converted_keys-rpdesc,
          ls_rpdesc  LIKE LINE OF lr_rpdesc,
          lr_product LIKE RANGE OF ls_converted_keys-product,
          ls_product LIKE LINE OF lr_product.
* Get filter or select option information
    lo_filter = io_tech_request_context->get_filter( ).
    lt_filter_select_options = lo_filter->get_filter_select_options( ).
    lv_filter_str = lo_filter->get_filter_string( ).
    IF lt_filter_select_options IS NOT INITIAL.
      LOOP AT lt_filter_select_options INTO ls_filter.
        CASE ls_filter-property.
          WHEN 'RPDESC'.
            lo_filter->convert_select_option(
              EXPORTING
                is_select_option = ls_filter
              IMPORTING
                et_select_option = lr_rpdesc ).
          WHEN 'PRODUCT'.
```

```
            lo_filter->convert_select_option(
            EXPORTING
              is_select_option = ls_filter
            IMPORTING
              et_select_option = lr_product ).
          WHEN OTHERS.
        ENDCASE.
      ENDLOOP.
    ENDIF.
**Get Material Master Release Package data from CDS View ZASV_MATMAS_RPG
    SELECT matnr_rpg
           rpg_desc
           matnr
           maktx
           prdha
           FROM zasv_matmas_rpg
           INTO TABLE lt_final_tab
           WHERE rpg_desc  IN lr_rpdesc
           AND   matnr     IN lr_product.
    IF sy-subrc = 0.
      LOOP AT lt_final_tab ASSIGNING FIELD-SYMBOL(<fs_final>).
        IF NOT <fs_final>-prdha IS INITIAL.
**<< Begin of comment by vijay on 09/03/2021 - To incorporate new prod hier design
*          <fs_final>-prodh1 = <fs_final>-prdha+0(5).
*          <fs_final>-prodh2 = <fs_final>-prdha+0(9).
*          <fs_final>-prodh3 = <fs_final>-prdha+0(14).
*          <fs_final>-prodh4 = <fs_final>-prdha+0(18).
**>> End of comment by vijay on 09/03/2021
**<< Begin of insert by vijay on 09/03/2021 - To incorporate new prod hier design
          <fs_final>-prodh1 = <fs_final>-prdha+0(2).
          <fs_final>-prodh2 = <fs_final>-prdha+0(5).
          <fs_final>-prodh3 = <fs_final>-prdha+0(9).
          <fs_final>-prodh4 = <fs_final>-prdha+0(14).
**>> End of insert by vijay on 09/03/2021
        ENDIF.
      ENDLOOP.
    ENDIF.
    IF lt_final_tab[] IS NOT INITIAL.
**Get the product hierarhy description
      SELECT prodh,
             vtext
             INTO TABLE @DATA(lt_suite)
             FROM t179t
             FOR ALL ENTRIES IN @lt_final_tab
             WHERE prodh EQ @lt_final_tab-prodh1.
      IF sy-subrc = 0.
      ENDIF.
      SELECT prodh,
             vtext
             INTO TABLE @DATA(lt_sol_family)
             FROM t179t
             FOR ALL ENTRIES IN @lt_final_tab
             WHERE prodh EQ @lt_final_tab-prodh2.
      IF sy-subrc = 0.
```

```
      ENDIF.
      SELECT prodh,
             vtext
             INTO TABLE @DATA(lt_solution)
             FROM t179t
             FOR ALL ENTRIES IN @lt_final_tab
             WHERE prodh EQ @lt_final_tab-prodh3.
      IF sy-subrc = 0.
      ENDIF.
      SELECT prodh,
             vtext
             INTO TABLE @DATA(lt_capability)
             FROM t179t
             FOR ALL ENTRIES IN @lt_final_tab
             WHERE prodh EQ @lt_final_tab-prodh4.
      IF sy-subrc = 0.
      ENDIF.
*  determine skip and top parameters
*  ---------------------------------------------------------------------
*        IF is_paging-skip IS NOT INITIAL.
*          lv_skip = is_paging-skip + 1.
*        ENDIF.
*
*        IF is_paging-top <> 0
*        AND lv_skip IS NOT INITIAL.
*          lv_top = is_paging-top + lv_skip - 1.
*        ELSEIF is_paging-top <> 0
*        AND lv_skip IS INITIAL.
*          lv_top = is_paging-top.
*        ELSE.
*          lv_top = lines( lt_final_tab ).
*        ENDIF.
      REFRESH: et_entityset.
      SORT lt_final_tab BY rpg_desc matnr.
      UNASSIGN: <fs_final>.
      LOOP AT lt_final_tab ASSIGNING <fs_final>.     "FROM lv_skip TO lv_top.
**  Get Suite Description
        READ TABLE lt_suite INTO DATA(ls_suite) WITH KEY prodh = <fs_final>-prodh1.
        IF sy-subrc = 0.
          <fs_final>-vtext1 = ls_suite-vtext.
        ENDIF.
**  Get Solution Family Description
        READ TABLE lt_sol_family INTO DATA(ls_sol_family) WITH KEY prodh =
<fs_final>-prodh2.
        IF sy-subrc = 0.
          <fs_final>-vtext2 = ls_sol_family-vtext.
        ENDIF.
**  Get Solution Description
        READ TABLE lt_solution INTO DATA(ls_solution) WITH KEY prodh =
<fs_final>-prodh3.
        IF sy-subrc = 0.
          <fs_final>-vtext3 = ls_solution-vtext.
        ENDIF.
**  Get Capability Description
```

```
        READ TABLE lt_capability INTO DATA(ls_capability) WITH KEY prodh =
<fs_final>-prodh4.
        IF sy-subrc = 0.
          <fs_final>-vtext4 = ls_capability-vtext.
        ENDIF.
**  Move the values to entity set
        ls_entity-rel_pkg = <fs_final>-matnr_rpg.
        ls_entity-rpdesc = <fs_final>-rpg_desc.
        ls_entity-product = <fs_final>-matnr.
        ls_entity-material_descr = <fs_final>-maktx.
        ls_entity-suite = <fs_final>-vtext1.
        ls_entity-solution_family = <fs_final>-vtext2.
        ls_entity-solution = <fs_final>-vtext3.
        ls_entity-capability = <fs_final>-vtext4.
        APPEND  ls_entity TO et_entityset.
        CLEAR: ls_entity.
      ENDLOOP.
    ENDIF.
    SORT et_entityset BY rel_pkg rpdesc product.
** $inlinecount query option for all count entries.
    IF io_tech_request_context->has_inlinecount( ) = abap_true.
      DESCRIBE TABLE et_entityset LINES es_response_context-inlinecount.
    ELSE.
      CLEAR: es_response_context-inlinecount.
    ENDIF.
**Sorting
    lt_order = io_tech_request_context->get_orderby( ).
    LOOP AT lt_order INTO ls_order.
      ls_oline-name = ls_order-property.
      IF ls_order-order = /iwbep/cl_mgw_data_util=>gcs_sorting_order-descending.
        ls_oline-descending = abap_true.
      ENDIF.
      APPEND ls_oline TO lt_otab.
      CLEAR ls_oline.
    ENDLOOP.
    IF lt_otab[] IS NOT INITIAL.
      SORT et_entityset BY (lt_otab).
    ENDIF.
*** The function module for $top and $skip Query Options
    CALL METHOD /iwbep/cl_mgw_data_util=>paging
      EXPORTING
        is_paging = is_paging
      CHANGING
        ct_data   = et_entityset.
  ENDMETHOD.
```

RPNAMESET_GET_ENTITYSET

```
  METHOD rpnameset_get_entityset.
**TRY.
*CALL METHOD SUPER->RPNAMESET_GET_ENTITYSET
*  EXPORTING
*    IV_ENTITY_NAME           =
*    IV_ENTITY_SET_NAME       =
```

```
*    IV_SOURCE_NAME           =
*    IT_FILTER_SELECT_OPTIONS =
*    IS_PAGING                =
*    IT_KEY_TAB               =
*    IT_NAVIGATION_PATH       =
*    IT_ORDER                 =
*    IV_FILTER_STRING         =
*    IV_SEARCH_STRING         =
**    io_tech_request_context  =
**  IMPORTING
**    et_entityset             =
**    es_response_context      =
*    .
** CATCH /iwbep/cx_mgw_busi_exception .
** CATCH /iwbep/cx_mgw_tech_exception .
**ENDTRY.
**Internal table and Workarea declaration
    DATA: lo_filter TYPE  REF TO /iwbep/if_mgw_req_filter.
    DATA: lt_filter_select_options TYPE /iwbep/t_mgw_select_option.
    DATA: ls_filter TYPE /iwbep/s_mgw_select_option.
    DATA: lv_filter_str TYPE string.
    DATA: ls_converted_keys LIKE LINE OF et_entityset.
    DATA: ls_entityset LIKE LINE OF et_entityset.
**Structure and Internal table for input parameters
    DATA: lr_rp_desc LIKE RANGE OF ls_converted_keys-zas_rp_desc,
          ls_rp_desc LIKE LINE OF lr_rp_desc.
**Get filter or select option information
    lo_filter = io_tech_request_context->get_filter( ).
    lt_filter_select_options = lo_filter->get_filter_select_options( ).
    lv_filter_str = lo_filter->get_filter_string( ).
    IF lt_filter_select_options[] IS NOT INITIAL.
      LOOP AT lt_filter_select_options INTO ls_filter.
        CASE ls_filter-property.
          WHEN 'ZAS_RP_DESC'.
            lo_filter->convert_select_option(
              EXPORTING
                is_select_option = ls_filter
              IMPORTING
                et_select_option = lr_rp_desc ).
          WHEN OTHERS.
        ENDCASE.
      ENDLOOP.
    ENDIF.
**Get Release Package Name details from CDS View ZCDSV_RPG_DATA
    SELECT rpg_desc
           FROM zasv_rpg_data
           INTO TABLE @DATA(lt_rpg_desc)
           WHERE rpg_desc IN @lr_rp_desc.
    IF sy-subrc = 0.
      SORT lt_rpg_desc BY rpg_desc.
      et_entityset[] = lt_rpg_desc[].
    ENDIF.
  ENDMETHOD.
```

RPVERSIONMASTERS_GET_ENTITYSET

```
  METHOD rpversionmasters_get_entityset.
**TRY.
*CALL METHOD SUPER->RPVERSIONMASTERS_GET_ENTITYSET
*  EXPORTING
*    IV_ENTITY_NAME           =
*    IV_ENTITY_SET_NAME       =
*    IV_SOURCE_NAME           =
*    IT_FILTER_SELECT_OPTIONS =
*    IS_PAGING                =
*    IT_KEY_TAB               =
*    IT_NAVIGATION_PATH       =
*    IT_ORDER                 =
*    IV_FILTER_STRING         =
*    IV_SEARCH_STRING         =
**    io_tech_request_context  =
**  IMPORTING
**    et_entityset             =
**    es_response_context      =
*    .
** CATCH /iwbep/cx_mgw_busi_exception .
** CATCH /iwbep/cx_mgw_tech_exception .
**ENDTRY.
**Internal table and Workarea declaration
    DATA: lo_filter TYPE  REF TO /iwbep/if_mgw_req_filter.
    DATA: lt_filter_select_options TYPE /iwbep/t_mgw_select_option.
    DATA: ls_filter TYPE /iwbep/s_mgw_select_option.
    DATA: lv_filter_str TYPE string.
    DATA: ls_converted_keys LIKE LINE OF et_entityset.
    DATA: ls_entityset LIKE LINE OF et_entityset.
    DATA: lt_order TYPE /iwbep/t_mgw_tech_order.
    DATA: ls_order     TYPE /iwbep/s_mgw_tech_order.
    DATA: lt_otab  TYPE abap_sortorder_tab,
          ls_oline TYPE abap_sortorder.
**Structure and Internal table for input parameters
    DATA: lr_rel_pkg_name LIKE RANGE OF ls_converted_keys-zas_rp_desc,
          ls_rel_pkg_name LIKE LINE OF lr_rel_pkg_name.
**Get filter or select option information
    lo_filter = io_tech_request_context->get_filter( ).
    lt_filter_select_options = lo_filter->get_filter_select_options( ).
    lv_filter_str = lo_filter->get_filter_string( ).
    IF lt_filter_select_options[] IS NOT INITIAL.
      LOOP AT lt_filter_select_options INTO ls_filter.
        CASE ls_filter-property.
          WHEN 'ZAS_RP_DESC'.
            lo_filter->convert_select_option(
              EXPORTING
                is_select_option = ls_filter
              IMPORTING
                et_select_option = lr_rel_pkg_name ).
          WHEN OTHERS.
        ENDCASE.
      ENDLOOP.
```

```
    ENDIF.
    SELECT zas_release_pkg,
           zas_rp_desc,
           zas_rp_version,
           zas_serv_flag,
           zas_n_rank_int,
           zas_ga_date,
           created_by,
           created_on,
           created_at,
           changed_by,
           changed_on,
           changed_at
           FROM zas_rp_vrsn_mast
           INTO TABLE @DATA(lt_rp_vrsn_mast)
           WHERE zas_rp_desc IN @lr_rel_pkg_name.
    IF sy-subrc = 0.
**Get the created name by passing the id from USER_ADDR table
      SELECT bname,
             name_textc
             FROM user_addr
             INTO TABLE @DATA(lt_user_addr)
             FOR ALL ENTRIES IN @lt_rp_vrsn_mast
             WHERE bname EQ @lt_rp_vrsn_mast-created_by.
**Get the changed name by passing the id from USER_ADDR table
      SELECT bname,
              name_textc
              FROM user_addr
              APPENDING TABLE @lt_user_addr
              FOR ALL ENTRIES IN @lt_rp_vrsn_mast
              WHERE bname EQ @lt_rp_vrsn_mast-changed_by.
      IF lt_user_addr[] IS NOT INITIAL.
        SORT lt_user_addr BY bname name_textc.
      ENDIF.
      LOOP AT lt_rp_vrsn_mast INTO DATA(ls_rp_vrsn_mast).
        IF ls_rp_vrsn_mast-zas_n_rank_int = '00'.
          ls_entityset-zas_n_rank = 'N'.
        ELSEIF ls_rp_vrsn_mast-zas_n_rank_int = '01'.
          ls_entityset-zas_n_rank = 'N-1'.
        ELSEIF ls_rp_vrsn_mast-zas_n_rank_int = '02'.
          ls_entityset-zas_n_rank = 'N-2'.
        ELSEIF ls_rp_vrsn_mast-zas_n_rank_int GE '03'.
          ls_entityset-zas_n_rank = 'N-9'.
        ENDIF.
        ls_entityset-zas_release_pkg = ls_rp_vrsn_mast-zas_release_pkg.
        ls_entityset-zas_rp_desc     = ls_rp_vrsn_mast-zas_rp_desc.
        ls_entityset-zas_rp_version  = ls_rp_vrsn_mast-zas_rp_version.
        ls_entityset-zas_serv_flag   = ls_rp_vrsn_mast-zas_serv_flag.
        ls_entityset-zas_ga_date     = ls_rp_vrsn_mast-zas_ga_date.
**Move Created by Name
        READ TABLE lt_user_addr INTO DATA(ls_user_addr) WITH KEY bname =
ls_rp_vrsn_mast-created_by
                                                                BINARY SEARCH.
        IF sy-subrc = 0.
```

```
          ls_entityset-created_by  = ls_user_addr-name_textc.
        ENDIF.
        ls_entityset-created_on      = ls_rp_vrsn_mast-created_on.
        ls_entityset-created_at      = ls_rp_vrsn_mast-created_at.
**Move Changed by Name
        CLEAR: ls_user_addr.
        READ TABLE lt_user_addr INTO ls_user_addr WITH KEY bname =
ls_rp_vrsn_mast-changed_by
                                                          BINARY SEARCH.
        IF sy-subrc = 0.
          ls_entityset-changed_by  = ls_user_addr-name_textc.
        ENDIF.
        ls_entityset-changed_on      = ls_rp_vrsn_mast-changed_on.
        ls_entityset-changed_at      = ls_rp_vrsn_mast-changed_at.
        APPEND ls_entityset TO et_entityset.
        CLEAR: ls_entityset,
               ls_user_addr.
      ENDLOOP.
*Begin of change by Prajeetha on 28/04/2022
*      Sort stmt changed : charm ID: 7000004548
*      SORT et_entityset BY  zas_rp_desc zas_n_rank.
       SORT et_entityset BY  zas_rp_desc zas_n_rank ASCENDING zas_ga_date DESCENDING.
*End of change by Prajeetha on 28/04/2022
** $inlinecount query option for all count entries.
      IF io_tech_request_context->has_inlinecount( ) = abap_true.
        DESCRIBE TABLE et_entityset LINES es_response_context-inlinecount.
      ELSE.
        CLEAR: es_response_context-inlinecount.
      ENDIF.
**Sorting
      lt_order = io_tech_request_context->get_orderby( ).
      LOOP AT lt_order INTO ls_order.
        ls_oline-name = ls_order-property.
        IF ls_order-order = /iwbep/cl_mgw_data_util=>gcs_sorting_order-descending.
          ls_oline-descending = abap_true.
        ENDIF.
        APPEND ls_oline TO lt_otab.
        CLEAR ls_oline.
      ENDLOOP.
      IF lt_otab[] IS NOT INITIAL.
        SORT et_entityset BY (lt_otab).
      ENDIF.
*** The function module for $top and $skip Query Options
      CALL METHOD /iwbep/cl_mgw_data_util=>paging
        EXPORTING
          is_paging = is_paging
        CHANGING
          ct_data   = et_entityset.
***To fix the sorting issue, changing the N-9 rank to >N-2 rank after sorting
      LOOP AT et_entityset ASSIGNING FIELD-SYMBOL(<fs_entityset>) WHERE zas_n_rank =
'N-9'.
        <fs_entityset>-zas_n_rank = '>N-2'.
      ENDLOOP.
    ENDIF.
```

```
  ENDMETHOD.
```

Local Types

```
*"* use this source file for any type of declarations (class
*"* definitions, interfaces or type declarations) you need for
*"* components in the private section
```

Local class definitions

```
*"* use this source file for the definition and implementation of
*"* local helper classes, interface definitions and type
*"* declarations
```

Macros

```
*"* use this source file for any macro definitions you need
*"* in the implementation part of the class
```

Overview