```
CALL METHOD cl_gui_cfw=>flush.

  data lr_utility TYPE REF TO ZDE_CL_CON_CPM_PW_UTILITY.
  lr_utility = ZDE_CL_CON_CPM_PW_UTILITY=>get_instance( ).

  lr_utility->gv_active_from = '20240301'.
  lr_utility->gv_active_to = '20240501'.
  CALL METHOD lr_utility->append_props_to_internal_plan.
  SELECT posid,post1 FROM prps INTO TABLE @DATA(lt_prps) UP TO 1 ROWS.
  IF sy-subrc = 0.
    MOVE-CORRESPONDING lt_prps TO et_entityset.
  ENDIF.
```

Message Class

/CPD/PFP_MESSAGES

Enhancement classess
zde_cl_con_cpm_ps_object_api=>gc_cpm_wbs1

Pls referring to the last column new naming convention

Thursday, April 11, 2024 10:35 AM

Object Type	**Name**	**Example**	**Naming to be Used.**
Enhancement Implementation for Class	Z+<Work Stream > +<Enhancement Category> + _+ <Meaningful Name >	ZCONEI_INVOICE_CHECK	ZCON_CPM_EI_INV_CHK
Enhancement Business Object	Z+<Work Stream>+_+EBO_+ <Meaningful Name >	ZCON_EBO_MP	ZCON_CPM_EBO_<The name of the enhancement object>
Data structure	Z+<Work Stream>+<S>_<Meaningful Name>	ZCONS_RPM_FIDATA	ZCON_CPM_DS_<The name of the Data Structure>
Transient Structure	Z+<Work Stream>+<TS>_<Meaningful Name>	ZCONTS_RPM_FIDATA	ZCON_CPM_TS_<The name of the Transient Structure>
Combined structure	Z+<Work Stream>+<CS>_<Meaningful Name>	ZCONCS_RPM_FIDATA	ZCON_CPM_CS_<The name of the Combined Structure>
Combined table type	Z+<Work Stream>+<TT>_<Meaningful Name >	ZCONTT_RPM_FIDATA	ZCON_CPM_CTT_<Combined Table Type>
Node name	Z+<Meaningful Name>	ZBUDGET	ZBGT_CPM
Determination Name	<Node Name>+<D>_<Meaningful Name>	ZBUDGETD_SETID	ZBGT_CPM_DT_<Determination Name>
Validation Name	<Node Name>+<V>_<Meaningful Name>	ZBUDGETV_MANDATORY_CHECK	ZBGT_CPM_VN_CHK_<Validation Name>
Action Name	<Node Name>+<A>_<Meaningful Name>	ZBUDGETA_CREATE_SNAPSHOT	ZBGT_CPM_AN_CHK_<Action Name>
Query Name	<Node Name>+<Q>_<Meaningful Name>	ZBUDGETQ_SELECT_ALL	ZBGT_CPM_QN_CHK_<Query Name>
Alternative Keys Name	<Node Name>+<KEY>	ZBUDGET_KEY	ZBGT_CPM_AKN_CHK_< Alternative Keys Name>
Enhancement Implementation for FPM UI	Z+<Work Stream > +<Enhancement Category> + _+ <Meaningful Name >	ZCONEI_PWS_WS_MP_HDR_NEW_FORM	ZCON_CPM_EI_FPM_UI_ <UI Name>

Mf4,9AhT8YxQ

```
 METHOD APPEND_PROPS_TO_INTERNAL_PLAN.
*   DATA:
*     lr_entity_type TYPE REF TO /iwbep/if_mgw_odata_entity_typ,
*     lr_entity_set  TYPE REF TO /iwbep/if_mgw_odata_entity_set,
**
*     lr_property    TYPE REF TO /IWBEP/IF_MGW_ODATA_property.
*   DATA:
*     lv_property_name TYPE /iwbep/med_external_name,
*     lv_months       TYPE i,
*     lv_period       TYPE num6,
*     lv_date         TYPE datum,
*     lv_period_from   TYPE num6,
*     lv_period_to     TYPE num6.
*
*   CONSTANTS:
*     lc_data_element_dec TYPE string VALUE 'ZCPM_PLAN_VALUE',
*      lc_prefix_plan_rate    TYPE string VALUE 'PRAT',
*     lc_prefix_plan_effort  TYPE string VALUE 'PEFT',
*     lc_prefix_plan_cost    TYPE string VALUE 'PCST'.
*   check model is BOUND.
*   lr_entity_type = model->get_entity_type( iv_entity_name = zcl_zcon_cpm_odata_pw__mpc=>
gc_internalplan ) ##NO_TEXT.
*   IF lr_entity_type IS BOUND.
*     lv_period_from = gv_active_from(6).
*     lv_period_to = gv_active_to(6).
*
*     lv_period = lv_period_from.
*
*     WHILE lv_period < lv_period_to.
*
*       CALL FUNCTION 'MONTH_PLUS_DETERMINE'
*         EXPORTING
*           months  = lv_months
*           olddate = gv_active_from  "active date
*         IMPORTING
*           newdate = lv_date.
*       lv_months = lv_months + 1.
*       lv_period = lv_date(6).
** Plan rate at Month
*       lv_property_name = lc_prefix_plan_rate && lv_period.
*       lr_property = lr_entity_type->create_property( iv_property_name = lv_property_name ).
*       lr_property->set_type_edm_decimal( ).
*       lr_property->set_is_key( ).
*       lr_property->set_nullable( abap_false ).
*       TRY.
*           CALL METHOD lr_property->bind_data_element
*             EXPORTING
*               iv_element_name = lc_data_element_dec
**              iv_bind_conversions = ABAP_FALSE
*             .
*         CATCH /iwbep/cx_mgw_med_exception.
```

```
*       ENDTRY.
** Plan cost at month
*       lv_property_name = lc_prefix_plan_cost && lv_period.
*       lr_property = lr_entity_type->create_property( iv_property_name = lv_property_name ).
*       lr_property->set_type_edm_decimal( ).
*       lr_property->set_is_key( ).
*       lr_property->set_nullable( abap_false ).
*       TRY.
*          CALL METHOD lr_property->bind_data_element
*            EXPORTING
*              iv_element_name = lc_data_element_dec
**              iv_bind_conversions = ABAP_FALSE
*            .
*        CATCH /iwbep/cx_mgw_med_exception.
*       ENDTRY.
** Plan effort at month
*       lv_property_name = lc_prefix_plan_effort && lv_period.
*       lr_property = lr_entity_type->create_property( iv_property_name = lv_property_name ).
*       lr_property->set_type_edm_decimal( ).
*       lr_property->set_is_key( ).
*       lr_property->set_nullable( abap_false ).
*       TRY.
*          CALL METHOD lr_property->bind_data_element
*            EXPORTING
*              iv_element_name = lc_data_element_dec
*            .
*        CATCH /iwbep/cx_mgw_med_exception.
*       ENDTRY.
*       IF lv_months > gc_months_max_number.
*         EXIT.
*       ENDIF.
*     ENDWHILE.
*   ENDIF.
 ENDMETHOD.
```

```
WHEN cl_fpm_event=>gc_event_close_dialog_box. " 'FPM_CLOSE_DIALOG'.
 iv_eventid->mo_event_data->get_value(              "Code for confirmation of copy version
       EXPORTING
        iv_key   = 'DIALOG_BOX_ID'
       IMPORTING
        er_value = lr_value ).
ASSIGN lr_value->* TO <fs_str>.
     IF <fs_str> = 'PAGE_8'.
 iv_eventid->mo_event_data->get_value(
             EXPORTING
              iv_key   = 'DIALOG_BUTTON_ACTION'
             IMPORTING
              er_value = lr_value
           ).
       ASSIGN lr_value->* TO <fs_str>.
       IF <fs_str> = 'OK'.
 lo_fpm->mo_app_parameter->get_value(
            EXPORTING
             iv_key   = 'POPUP_TYPE'
            IMPORTING
             ev_value = lv_popup_type
          ).
         lo_fpm->mo_app_parameter->get_value(
          EXPORTING
           iv_key   = 'FORECAST_PERIOD'
          IMPORTING
           ev_value = lv_fc_period
        ).

endif.
```

```
 lv_fc_period      TYPE          /cpd/pfp_forecast_period,
 lo_fpm->mo_app_parameter->set_value(
       EXPORTING
         iv_key   = 'PLAN_PERIOD'
         iv_value = lv_plan_period
     ).
 lt_version_detail  TYPE /cpd/t_pfp_plan_version,
       ls_version_detail  TYPE /cpd/s_pfp_plan_version,
lv_version        TYPE /cpd/pfp_version_id,
lv_forecast_month     TYPE /cpd/pfp_plan_forecast_period,
        lv_forecast_year      TYPE /cpd/pfp_plan_forecast_year,
* Check for snapshot creation
     lv_forecast_month = lv_forecast_period+1(2).
     lv_forecast_year = lv_forecast_period+4(4).
```

```
pspnr,
    posid,
    usr00,
    usr02,
    usr03,
    usr04,
    usr09,
    usr10,
    zzext_cooper,
    zzcooper_detail,
    zzboard_name,
    zzboard_date,
    zzcustomer_vis,
    zzcontract_typ
```

```
 METHOD get_result_data.
*---------------------------------------------------------------------------------------------------------------
*  Author               : Baobao Gao – DTSRB2T
*  Functional Owner     : Yanming Xu - AHOR2ME
*  Date                 : 17/05/2024
*  SAP Charm # / JIRA ID #  : CPMP-56
*  Description          : Multi Project Report
*  RICEFW ID            : CPM011
*---------------------------------------------------------------------------------------------------------------
*---------------------------------------------------------------------------------------------------------------
*  Author               : Baobao Gao – DTSRB2T
*  Functional Owner     : Yanming Xu - AHOR2ME
*  Date                 : 17/05/2024
*  SAP Charm # / JIRA ID #  : CPMP-56
*  Description          : Add Fields (Projectuuid/PlanId/VersionId)
*  Workbench            : V1DK905918
*---------------------------------------------------------------------------------------------------------------

  DATA:
   lv_acc_spec_id  TYPE /cpd/auth_acc_spec_id,
   lt_access_type  TYPE /cpd/auth_t_acc_spec_id,
   lt_authorize    TYPE /cpd/auth_t_acc_spec_list,
   ls_authorize    TYPE /cpd/auth_s_acc_spec_list,
   ls_result          TYPE zcon_cpms_mp_multi_result,
   ls_data             TYPE zcon_cpms_mp_multi_result,
   lt_data             TYPE zcon_cpmt_mp_multi_result,
   lt_plan       TYPE  TABLE OF zcon_cpms_pw_cube_r01_plan,
   lv_tabix      TYPE i,
   lt_month_names  TYPE STANDARD TABLE OF t247,
   ls_month_names  TYPE t247,
   ls_plan       TYPE  zcon_cpms_pw_cube_r01_plan,
   lv_pua_id     TYPE /cpd/pws_ws_bu_partner,
   lv_period_from  TYPE zereportingperiod,
   lv_period_to    TYPE zereportingperiod,
   lv_search_value TYPE  string VALUE '*',
   lt_data_bp     TYPE /cpd/cl_pws_ws_ui_utility=>tt_result_table,
   ls_result_bp    TYPE /cpd/cl_pws_ws_ui_utility=>ts_result_table.

  FIELD-SYMBOLS:
   <lt_plan> TYPE ANY TABLE,
   <ls_plan> TYPE any,
   <ls_data> TYPE zcon_cpms_mp_multi_result.
  DATA:
   lr_projectid             TYPE RANGE OF zeprojectid,
   lr_wbsid                 TYPE RANGE OF ps_posid,
   lr_projectstatus         TYPE RANGE OF zeprojectstatus,
   lr_serviceorganization     TYPE RANGE OF zeserviceorganization,
   lr_createdby             TYPE RANGE OF /bofu/user_id_created_by,
   lr_projectcontroller       TYPE RANGE OF zeprojectcontroller,
   lr_projectmanager         TYPE RANGE OF zeprojectmanager,
   lr_planningfield          TYPE RANGE OF zeplanningfield,
   lr_subplanningfield       TYPE RANGE OF zesubplanningfield,
   lr_techcentercentralenabler TYPE RANGE OF zetechcentercentralenabler,
   lr_subsystem             TYPE RANGE OF zesubsystem,
   lr_externalcooperation     TYPE RANGE OF zext_cooper,
```

```
  lr_cooperationdetail       TYPE RANGE OF zcooper_detail,
  lr_kostl                   TYPE RANGE OF kostl,
  lr_lifnr                   TYPE RANGE OF lifnr,
  lv_reportingtype           TYPE  zereportingtype,
  lv_reportingview           TYPE  zereportingview,
  lv_reportingperiod         TYPE  zereportingperiod.
DATA:
  lr_bukrs       TYPE RANGE OF ps_pbukr,
  ls_input_commit TYPE zfm_commit_in_s,
  ls_input_actual TYPE zfm_actual_in_s,
  ls_input_budget TYPE zfm_bgtdoc_in_s,
  lt_budget_tra   TYPE zfm_bgtdoc_out_t,
  lt_budget_rel   TYPE zfm_bgtdoc_out_t,
  ls_budget      TYPE zfm_bgtdoc_out_s,
  lt_actual      TYPE zfm_actual_out_t,
  ls_actual      TYPE zfm_actual_out_s,
  lt_commit      TYPE zfm_commit_out_t,
  ls_commit      TYPE zfm_commit_out_s
  .
DATA :
  lt_keyfig  TYPE     rsd_t_dta_pro,
  lt_char    TYPE     rsd_t_dta_pro,
  lr_plan    TYPE     REF TO  data,
  lt_message TYPE     bsanly_t_message,
  ls_range   TYPE     rsdri_s_range,
  lt_range   TYPE     rsdri_t_range.
CONSTANTS:
  lc_bprole                TYPE bu_partnerrole VALUE 'BUP003',
  lc_forecast_status_0ini     TYPE /cpd/pfp_forecast_st_id VALUE '0INI',
  lc_forecast_status_0inp     TYPE /cpd/pfp_forecast_st_id VALUE '0INP',
  lc_forecast_status_0com     TYPE /cpd/pfp_forecast_st_id VALUE '0COM',
  lc_uom                   TYPE  msehi VALUE 'H',
  lc_budtype                 TYPE  char4 VALUE 'RL00',
  lc_reportingtype_proj       TYPE  zereportingtype VALUE 'T01',
  lc_reportingtype_prps       TYPE  zereportingtype VALUE 'T02',
  lc_reportingtype_ftyp       TYPE  zereportingtype VALUE 'T03',
  lc_reportingtype_fres       TYPE  zereportingtype VALUE 'T04',
  lc_reportingview_overview   TYPE  zereportingview VALUE 'V01',
  lc_reportingview_monthly    TYPE  zereportingview VALUE 'V02',
  lc_version_type_id         TYPE  /cpd/pfp_ver_type_id VALUE 'PLAN',
  lc_plan_structure          TYPE  /cpd/pfp_structure VALUE 'E',
  lc_domname_struct_sel      TYPE domname VALUE '/CPD/PFP_STRUCT_SEL',
  lc_projectid               TYPE char30 VALUE 'Projectid',
  lc_wbsid                   TYPE char30 VALUE 'Wbsid',
  lc_projectstatus           TYPE char30 VALUE 'Projectstatus',
  lc_serviceorganization     TYPE char30 VALUE 'Serviceorganization',
  lc_createdby               TYPE char30 VALUE 'Createdby',
  lc_projectcontroller       TYPE char30 VALUE 'Projectcontroller',
  lc_projectmanager          TYPE char30 VALUE 'Projectmanager',
  lc_planningfield           TYPE char30 VALUE 'Planningfield',
  lc_subplanningfield        TYPE char30 VALUE 'Subplanningfield',
  lc_techcentercentralenabler TYPE char30 VALUE 'Techcentercentralenabler',
  lc_subsystem               TYPE char30 VALUE 'Subsystem',
  lc_externalcooperation     TYPE char30 VALUE 'Externalcooperation',
  lc_cooperationdetail       TYPE char30 VALUE 'Cooperationdetail',
  lc_reportingtype           TYPE char30 VALUE 'Reportingtype',
  lc_reportingview           TYPE char30 VALUE 'Reportingview',
```

```
    lc_reportingperiod        TYPE char30 VALUE 'Reportingperiod'.

  lv_period_from = sy-datum(6).
  lv_period_to = sy-datum(6).
  IF it_selection_param[] IS INITIAL.
*    ls_mess-msgid      = 'ZCON_CPM'.
*    ls_mess-msgno      = 000.
*    ls_mess-severity   = 'W'.
*    ls_mess-parameter_1   = 'it takes long time without selection parameter.'.
*    APPEND ls_mess TO et_messages.
  ENDIF.
*  Get Project Stage Description
  SELECT DISTINCT engagementprojectstage AS stage, engagementprojectstagetext AS description
FROM i_engagementprojectstagetext AS status INTO TABLE  @DATA(lt_projstage) WHERE language
= @sy-langu.
*  Get Resource Type Description
  SELECT DISTINCT res_type_id,description FROM zcon_cpm_ddl_i_pwrtyp INTO TABLE
@DATA(lt_restyp).
  IF sy-subrc = 0.
    SORT lt_restyp BY res_type_id.

*  Based on Resource Type's UoM
    SELECT DISTINCT a~res_type_id, a~uom ,a~unitofmeasurelongname AS text
    INTO  TABLE @DATA(lt_uomt)
    FROM zcon_cpm_ddl_i_pwuom AS a      "#EC CI_SUBRC"#EC CI_NOWHERE
    FOR ALL ENTRIES IN @lt_restyp
    WHERE res_type_id = @lt_restyp-res_type_id.
    IF sy-subrc = 0.
     SORT lt_uomt BY res_type_id uom.
    ENDIF.

  ENDIF.
*  Based on Resource Type's Resource/Resource Description
  SELECT DISTINCT resource_id,description,res_type_id FROM zcon_cpm_ddl_i_pwres INTO TABLE
@DATA(lt_res).
*  Get the connection between Resource Type and Commitment Item
  SELECT DISTINCT                                "#EC CI_SUBRC
    a~res_type,
    a~fipex
  INTO TABLE @DATA(lt_fipex)
  FROM zconv_fm_res_typ AS a
  INNER JOIN /cpd/fc_rty AS b ON a~res_type = b~res_type_id.
  IF sy-subrc = 0.
    SORT lt_fipex BY res_type fipex.
    DELETE ADJACENT DUPLICATES FROM lt_fipex COMPARING ALL FIELDS.
  ENDIF.

  CALL FUNCTION 'MONTH_NAMES_GET'
    EXPORTING
     language    = 'E'
    TABLES
     month_names = lt_month_names.
  "   Forecast status text
  SELECT DISTINCT EngagementProjectReviewStatus AS status, EngagementProjReviewStatusText
AS text FROM i_engmntprojfinplanstatustext INTO TABLE @DATA(lt_forecast_status_desc)
    WHERE Language = @sy-langu.
*  Convert Input Data
```

```
  LOOP AT it_selection_param INTO DATA(ls_selection_param).
   IF ls_selection_param-property EQ lc_projectid.
    lr_projectid = CORRESPONDING #( ls_selection_param-select_options ).
   ELSEIF ls_selection_param-property EQ lc_wbsid.
    lr_wbsid = CORRESPONDING #( ls_selection_param-select_options ).
   ELSEIF ls_selection_param-property EQ lc_projectstatus.
    lr_projectstatus = CORRESPONDING #( ls_selection_param-select_options ).
   ELSEIF ls_selection_param-property EQ lc_serviceorganization.
    lr_serviceorganization = CORRESPONDING #( ls_selection_param-select_options ).
   ELSEIF ls_selection_param-property EQ lc_createdby.
    lr_createdby = CORRESPONDING #( ls_selection_param-select_options ).
   ELSEIF ls_selection_param-property EQ lc_projectcontroller.
    lr_projectcontroller = CORRESPONDING #( ls_selection_param-select_options ).
   ELSEIF ls_selection_param-property EQ lc_projectmanager.
    lr_projectmanager = CORRESPONDING #( ls_selection_param-select_options ).
   ELSEIF ls_selection_param-property EQ lc_externalcooperation.
    lr_externalcooperation = CORRESPONDING #( ls_selection_param-select_options ).
   ELSEIF ls_selection_param-property EQ lc_cooperationdetail.
    lr_cooperationdetail = CORRESPONDING #( ls_selection_param-select_options ).
   ELSEIF ls_selection_param-property EQ lc_planningfield.
    lr_planningfield = CORRESPONDING #( ls_selection_param-select_options ).
   ELSEIF ls_selection_param-property EQ lc_subplanningfield.
    lr_subplanningfield = CORRESPONDING #( ls_selection_param-select_options ).
   ELSEIF ls_selection_param-property EQ lc_techcentercentralenabler.
    lr_techcentercentralenabler = CORRESPONDING #( ls_selection_param-select_options ).
   ELSEIF ls_selection_param-property EQ lc_subsystem.
    lr_subsystem = CORRESPONDING #( ls_selection_param-select_options ).
   ELSEIF ls_selection_param-property EQ lc_reportingtype.
    lv_reportingtype = ls_selection_param-select_options[ 1 ]-low.
   ELSEIF ls_selection_param-property EQ lc_reportingview.
    lv_reportingview = ls_selection_param-select_options[ 1 ]-low.
   ELSEIF ls_selection_param-property EQ lc_reportingperiod.
    IF ls_selection_param-select_options IS NOT INITIAL.
     lv_period_from = ls_selection_param-select_options[ 1 ]-low+0(6).
     lv_period_to = ls_selection_param-select_options[ 1 ]-low+7(6).
    ENDIF.
   ENDIF.
  ENDLOOP.

*  Get Project/team/reporting attributes Detail Based on Importing Filters
  SELECT DISTINCT
      hdr~mp_id AS projectid,
      hdr~created_by AS createdby,
      hdr~org_id,
      hdr~mp_type,
      hdr~mp_stage AS projectstatus,
      hdr~proj_mgr_bupa_id,
      hdr~project_type,
      hdr~start_date,
      hdr~end_date,
*       hdr~ext_cooper AS externalcooperation,
*       hdr~cooper_detail AS cooperationdetail,
      hdr~plan_field,
      hdr~sub_planfield,
      hdr~tech_center,
      hdr~sub_system,
      hdr~db_key,
```

```
        item~mp_itm_otyp,
        item~mp_item_okey,
        orgt~org_unit_desc AS serviceorganization,


        extcpnvh~externalcooperationname  AS externalcooperation,
        cpndetvh~cooperationdetailname  AS cooperationdetail,
        plfldt~planningfieldname  AS planningfield,
        subpft~subplanningfieldname  AS subplanningfield,
        tecctt~techcentercentralenablername  AS techcentercentralenabler,
        subsyt~subsystemname  AS subsystem,

        wbselementdata~wbselementinternalid,
        wbselementdata~wbselementexternalid AS wbsid,

        wbselementdata~wbselement,
        wbselementdata~wbsdescription AS wbsname,
        wbselementdata~wbselementobject,
        wbselementdata~controllingarea,
        wbselementdata~responsiblecostcenter AS respcostcenter,

        r_z001~engagementprojectteamrole AS croleid,
        controller~employmentinternalid  AS projectcontroller,

        r_z002~engagementprojectteamrole AS mroleid,
        manager~employmentinternalid  AS projectmanager,

        hdrt~text AS projectname
    FROM /cpd/d_mp_hdr AS hdr
    LEFT OUTER JOIN /cpd/pwsc_orgidt AS orgt ON hdr~org_id = orgt~org_unit_id AND orgt~spras =
@sy-langu
    LEFT OUTER JOIN /cpd/d_mp_hdr_s AS hdrt ON hdr~db_key = hdrt~parent_key
    LEFT OUTER JOIN zcon_cpm_ddl_c_extcpnvh AS extcpnvh ON hdr~ext_cooper =
extcpnvh~externalcooperation
    LEFT OUTER JOIN zcon_cpm_ddl_c_cpndetvh AS cpndetvh ON hdr~ext_cooper =
cpndetvh~externalcooperation AND hdr~cooper_detail = cpndetvh~cooperationdetail
    LEFT OUTER JOIN zcon_cpm_ddl_i_plfldt AS plfldt ON hdr~plan_field = plfldt~planningfield AND
plfldt~language = @sy-langu
    LEFT OUTER JOIN zcon_cpm_ddl_i_subpft AS subpft ON hdr~sub_planfield =
subpft~subplanningfield AND subpft~language = @sy-langu
    LEFT OUTER JOIN zcon_cpm_ddl_i_tecctt AS tecctt ON hdr~tech_center =
tecctt~techcentercentralenabler AND tecctt~language = @sy-langu
    LEFT OUTER JOIN zcon_cpm_ddl_i_subsyt AS subsyt ON hdr~sub_system = subsyt~subsystem
AND subsyt~language = @sy-langu
    LEFT OUTER JOIN /cpd/d_mp_item AS item ON hdr~db_key = item~parent_key AND
item~mp_itm_otyp = @zde_cl_con_cpm_ps_object_link=>gc_ps_obj_link_0wbs
    LEFT OUTER JOIN prps ON item~mp_item_okey = prps~objnr
    LEFT OUTER JOIN prhi ON prps~pspnr = prhi~up
    LEFT OUTER JOIN i_wbselementdata AS wbselementdata ON prhi~posnr =
wbselementdata~wbselementinternalid AND wbselementdata~wbselementhierarchylevel = 2
    LEFT OUTER JOIN i_engagementprojectteam AS team      ON hdr~db_key =
team~engagementprojectuuid
    LEFT OUTER JOIN i_engagementprojectteamrole AS r_z001  ON
team~engagementprojectteamuuid = r_z001~engagementprojectteamuuid AND r_z001
~engagementprojectteamrole = 'Z001'
    LEFT OUTER JOIN i_engmtprojteammember AS teammember1  ON r_z001
```

```
~engagementprojectteamroleuuid = teammember1~engagementprojectteamroleuuid
   LEFT OUTER JOIN i_engagementprojectmember AS controller  ON teammember1
~engagementprojectmemberuuid = controller~engagementprojectmemberuuid
   LEFT OUTER JOIN i_engagementprojectteamrole AS r_z002  ON
team~engagementprojectteamuuid = r_z002~engagementprojectteamuuid AND r_z002
~engagementprojectteamrole = 'Z002'
   LEFT OUTER JOIN i_engmtprojteammember AS teammember2  ON r_z002
~engagementprojectteamroleuuid = teammember2~engagementprojectteamroleuuid
   LEFT OUTER JOIN i_engagementprojectmember AS manager  ON teammember2
~engagementprojectmemberuuid = manager~engagementprojectmemberuuid
   WHERE hdr~mp_id IN @lr_projectid
   AND   hdr~mp_stage IN @lr_projectstatus
   AND   hdr~org_id IN @lr_serviceorganization
   AND   hdr~created_by IN @lr_createdby
   AND   controller~employmentinternalid IN @lr_projectcontroller
   AND   manager~employmentinternalid IN @lr_projectmanager
   AND   hdr~plan_field IN @lr_planningfield
   AND   hdr~sub_planfield IN @lr_subplanningfield
   AND   hdr~tech_center IN @lr_techcentercentralenabler
   AND   hdr~sub_system IN @lr_subsystem
   AND   hdr~ext_cooper IN @lr_externalcooperation
   AND   hdr~cooper_detail IN @lr_cooperationdetail
   AND   wbselementdata~WBSElementExternalID IN @lr_wbsid
   INTO TABLE @DATA(lt_proj).
  IF sy-subrc = 0.
*    Get Project's Owner
    SELECT DISTINCT userid AS createdby,username FROM zcon_cpm_ddl_c_mpuser FOR ALL
ENTRIES IN @lt_proj
      WHERE userid = @lt_proj-createdby
      INTO TABLE @DATA(lt_user).
*    Project Manager Description
    lr_projectmanager = VALUE #( FOR wa_proj IN lt_proj ( sign = 'I' option = 'EQ' low = wa_proj-
projectmanager ) ).
    SORT lr_projectmanager BY low.
    DELETE ADJACENT DUPLICATES FROM lr_projectmanager COMPARING low.
    SELECT DISTINCT businesspartner,businesspartnerfullname FROM i_mstrprojbpcontact INTO
TABLE @DATA(lt_mp_bp_pm)
    WHERE businesspartnerrole = @lc_bprole AND businesspartner IN @lr_projectmanager.
*    Project Controller Description
    lr_projectcontroller = VALUE #( FOR wa_proj IN lt_proj ( sign = 'I' option = 'EQ' low = wa_proj-
projectcontroller ) ).
    SORT lr_projectcontroller BY low.
    DELETE ADJACENT DUPLICATES FROM lr_projectcontroller COMPARING low.
    SELECT DISTINCT businesspartner,businesspartnerfullname FROM i_mstrprojbpcontact INTO
TABLE @DATA(lt_mp_bp_pc)
    WHERE businesspartnerrole = @lc_bprole AND businesspartner IN @lr_projectcontroller.

*    Import Company Code,Funds Center,Commitment Item,Fiscal Year Get Actual Quantity and
Actual Cost
*    Company Code
    lr_bukrs = VALUE #( FOR wa_proj IN lt_proj ( sign = 'I' option = 'EQ' low = wa_proj-
controllingarea ) ).
    SORT lr_bukrs BY low.
    DELETE ADJACENT DUPLICATES FROM lr_bukrs COMPARING low.
    DELETE lr_bukrs WHERE low IS INITIAL.
    ls_input_actual-fikrs  = lr_bukrs.
*    Funds Center
```

```
    ls_input_actual-fictr  = VALUE #( FOR wa_proj IN lt_proj ( sign = 'I' option = 'EQ' low = wa_proj-
wbsid ) ).
    SORT ls_input_actual-fictr BY low ASCENDING.
    DELETE ADJACENT DUPLICATES FROM ls_input_actual-fictr COMPARING low.
    DELETE ls_input_actual-fictr WHERE low IS INITIAL.
*    Commitment Item
    LOOP AT lt_fipex INTO DATA(ls_fipex).
     APPEND VALUE #( sign = 'I' option = 'EQ'  low = ls_fipex-fipex ) TO ls_input_actual-cmmtitem.
    ENDLOOP.
*    Fiscal Year
    APPEND VALUE #( sign = 'I' option = 'BT'  low = lv_period_from(4) high = lv_period_to(4) ) TO
ls_input_actual-gjahr.
*    Get Actual Quantity and Actual Cost
    IF ls_input_actual-fictr IS NOT INITIAL.
     CALL FUNCTION 'ZFM_DERIVE_ACTUAL'
      EXPORTING
       is_input  = ls_input_actual
      IMPORTING
       et_output = lt_actual.
     SORT lt_actual BY fistl.
*    Import Company Code,Funds Center,Commitment Item,Fiscal Year  Get Commitment.
     MOVE-CORRESPONDING ls_input_actual TO ls_input_commit.
     CALL FUNCTION 'ZFM_DERIVE_COMMIT'
      EXPORTING
       is_input  = ls_input_commit
      IMPORTING
       et_output = lt_commit.
     SORT lt_commit BY fistl.
     IF lv_reportingtype NE lc_reportingtype_fres.
*     Import Company Code,Funds Center,Commitment Item,Fiscal Year  Get Released Budget
      MOVE-CORRESPONDING ls_input_actual TO ls_input_budget.
      APPEND VALUE #( sign = 'I' option = 'EQ'  low = 'R1' ) TO ls_input_budget-valtype.
      CALL FUNCTION 'ZFM_GET_BUDGET_DETAIL'
       EXPORTING
        is_input  = ls_input_budget
       IMPORTING
        et_output = lt_budget_rel.
      SORT lt_budget_rel BY fundsctr.
*     Import Company Code,Funds Center,Commitment Item,Fiscal Year Get Transferred Budget
      CLEAR ls_input_budget-valtype.
      APPEND VALUE #( sign = 'I' option = 'EQ'  low = 'B1' ) TO ls_input_budget-valtype.
      CALL FUNCTION 'ZFM_GET_BUDGET_DETAIL'
       EXPORTING
        is_input  = ls_input_budget
       IMPORTING
        et_output = lt_budget_tra.
      SORT lt_budget_tra BY fundsctr.
*     Import Funds Center,Commitment Item,Fiscal Year Get  Target Budget
      SELECT DISTINCT gjahr,fictr,fipex,ztarget FROM zcont_fm_tar_val
       FOR ALL ENTRIES IN @lt_fipex WHERE fipex = @lt_fipex-fipex
       AND fictr IN @ls_input_budget-fictr
       AND gjahr IN @ls_input_budget-gjahr
       INTO TABLE @DATA(lt_budget_tar).
      IF sy-subrc = 0.
       SORT lt_budget_tar BY fictr.
      ENDIF.
     ENDIF.
```

```
    ENDIF.

*    Get Financial Plan & Version Information
    SELECT DISTINCT                    "#EC CI_FAE_LINES_ENSURED
      s2~plan_id,
      s2~external_id,
      s2~mp_id_int,
      s2~status_id AS plan_status,
      s2~sel_structure,
      s2~period_end_forecast_status,
      s2~forecast_period,
      s3~ddtext AS plan_hierarchy,
      s4~description AS fc_desc,

      s1~version_id,
      s1~description,
      s1~status_id,
      s1~plan_forecast_year,
      s1~plan_forecast_period,
      s1~version_type

      INTO TABLE @DATA(lt_pfp_pv)
      FROM /cpd/d_pfp_pv AS s1
      INNER JOIN /cpd/d_pfp_ph AS s2 ON s1~parent_key = s2~db_key
      LEFT JOIN dd07v AS s3  ON  s2~sel_structure = s3~domvalue_l AND s3~domname =
@lc_domname_struct_sel AND s3~ddlanguage = @sy-langu
      LEFT JOIN /cpd/fc_status_t AS s4  ON  s2~period_end_forecast_status = s4~status_id  AND s4
~spras = @sy-langu
      FOR ALL ENTRIES IN @lt_proj
      WHERE  s1~version_type = @lc_version_type_id
        AND  s2~mp_id_int = @lt_proj-db_key
        AND  s2~sel_structure = @lc_plan_structure
 .
    IF sy-subrc = 0.
*       Import Plan ID,Version ID, Calendar Month Get Plan Quantity and Plan Cost.
      LOOP AT lt_pfp_pv INTO DATA(ls_pfp_pv).
        ls_range-chanm  = /cpd/cl_pfp_constants=>gc_fpoid. "Name Of Characteristic
        ls_range-sign   = 'I'.             "Include
        ls_range-compop = 'EQ'.            "Operator
        ls_range-low    = ls_pfp_pv-plan_id.
        APPEND ls_range TO lt_range.
        CLEAR ls_range.

        ls_range-chanm  = /cpd/cl_pfp_constants=>gc_fver. "Name Of Characteristic
        ls_range-sign   = 'I'.             "Include
        ls_range-compop = 'EQ'.            "Operator
        ls_range-low    = ls_pfp_pv-version_id.
        APPEND ls_range TO lt_range.
        CLEAR ls_range.
      ENDLOOP.

      "Pass filter for Calendar Month
      ls_range-chanm  = /cpd/cl_pfp_constants=>gc_calmonth. "Name Of Characteristic
      ls_range-sign   = 'I'.             "Include
      ls_range-compop = 'BT'.            "Operator
      ls_range-low    = lv_period_from.
      ls_range-high   = lv_period_to.
```

```
      APPEND ls_range TO lt_range.
      CLEAR ls_range.
      SORT lt_range BY chanm sign compop low high.
      DELETE ADJACENT DUPLICATES FROM lt_range COMPARING chanm sign compop low high.

*     Get Plan Quantity and Plan Cost
      CALL FUNCTION 'ZDE_CON_CPM_FM_FP_READ_DATA'
       EXPORTING
        it_range = lt_range
       IMPORTING
        io_data  = lr_plan.

      ASSIGN lr_plan->* TO <lt_plan>.
      IF sy-subrc = 0.
       lt_plan = CORRESPONDING #( <lt_plan> ).
       SORT lt_plan BY /cpd/fpid
                       /cpd/fpoid
                       /cpd/fver
                       /cpd/frtyp
                       /cpd/fres
                       zcpm_vdor
                       /cpd/cstcnt
                       /cpd/ftcur
                       /cpd/fuom
                       0calmonth
                       .
       DELETE lt_plan WHERE /cpd/fqty = 0 AND /cpd/ftrate = 0 AND /cpd/ftca = 0 AND /cpd/fetca =
0.
      ENDIF.
     ENDIF.
    ENDIF.

    lv_acc_spec_id = /cpd/cl_common_access_constant=>gc_auth_mp_milestone_read.
    APPEND /cpd/cl_common_access_constant=>gc_auth_mp_milestone_read TO lt_access_type.

    LOOP AT lt_proj INTO DATA(ls_proj).
     CLEAR lt_authorize.
*    Verify permissions
     CALL FUNCTION '/CPD/AUTH_CHECK_MP_ACCESS'
      EXPORTING
       iv_mp_key     = ls_proj-db_key
       it_access_type = lt_access_type
      IMPORTING
       et_authorized  = lt_authorize.
     READ TABLE lt_authorize INTO ls_authorize WITH KEY acc_spec_id = lv_acc_spec_id.
     IF sy-subrc <> 0.
      CONTINUE.
     ENDIF.
     MOVE-CORRESPONDING ls_proj TO ls_data.
     ls_data-projectuuid = ls_proj-db_key."Project GUID
     READ TABLE lt_pfp_pv INTO ls_pfp_pv WITH KEY  mp_id_int = ls_proj-db_key.
     IF sy-subrc = 0.
      ls_data-planid = ls_pfp_pv-plan_id."Plan id
      ls_data-versionid = ls_pfp_pv-version_id."Version id
      IF ls_pfp_pv-plan_forecast_year > 0 AND ls_pfp_pv-plan_forecast_period > 0.
       "Forecast period
       ls_data-forecastperiod+0(4) = ls_pfp_pv-plan_forecast_year .
```

```
        ls_data-forecastperiod+4(2) = ls_pfp_pv-plan_forecast_period+1(2) .
        "Forecast status
        IF ls_pfp_pv-period_end_forecast_status NE lc_forecast_status_0ini.
          ls_data-forecaststatus = ls_pfp_pv-period_end_forecast_status.
        ENDIF.
        IF ls_data-forecaststatus IS INITIAL.
          ls_data-forecaststatus = lc_forecast_status_0inp.
        ENDIF.
      ENDIF.
    ENDIF.
*    Get project controller name
    IF ls_proj-projectcontroller IS NOT INITIAL.
      READ TABLE lt_mp_bp_pc INTO DATA(ls_mp_bp_pc) WITH KEY businesspartner = ls_proj-
projectcontroller.
      IF sy-subrc EQ 0.
        ls_data-projectcontroller = ls_mp_bp_pc-businesspartnerfullname.
      ENDIF.
    ENDIF.
*    Get project manager name
    IF ls_proj-projectmanager IS NOT INITIAL.
      READ TABLE lt_mp_bp_pm INTO DATA(ls_mp_bp_pm) WITH KEY businesspartner = ls_proj-
projectmanager.
      IF sy-subrc EQ 0.
        ls_data-projectmanager = ls_mp_bp_pm-businesspartnerfullname.
      ENDIF.
    ENDIF.

    READ TABLE lt_plan TRANSPORTING NO FIELDS WITH KEY /cpd/fpid = ls_proj-wbselementobject.
    IF sy-subrc = 0.
      lv_tabix = sy-tabix.
      "Plan cost/Plan quantity
      LOOP AT lt_plan INTO ls_plan FROM lv_tabix WHERE /cpd/fpid = ls_proj-wbselementobject.
        ls_data-calendarmonth =  ls_plan-0calmonth.
        ls_data-resourcetypeid = ls_plan-/cpd/frtyp.
        ls_data-resourceid = ls_plan-/cpd/fres.
        ls_data-uom = ls_plan-/cpd/fuom.
        ls_data-planquantity = ls_plan-/cpd/fqty.
        ls_data-plancost = ls_plan-/cpd/ftca.
        ls_data-costcenter = ls_plan-/cpd/cstcnt.
        ls_data-vendor = ls_plan-zcpm_vdor.
        IF ls_data-calendarmonth >= sy-datum(6).'> ls_data-forecastperiod.
          ls_data-etccost = ls_plan-/cpd/fetca."forecast etc
        ENDIF.
        COLLECT ls_data INTO lt_data.
        CLEAR:
          ls_data-calendarmonth,
          ls_data-resourcetypeid,
          ls_data-resourceid,
          ls_data-uom,
          ls_data-planquantity,
          ls_data-plancost,
          ls_data-costcenter,
          ls_data-vendor,
          ls_data-etccost
          .
      ENDLOOP.
    ENDIF.
```

```
READ TABLE lt_actual TRANSPORTING NO FIELDS WITH KEY fistl = ls_proj-wbsid.
IF sy-subrc = 0.
  lv_tabix = sy-tabix.
  "Actual cost/Actual quantity
  LOOP AT lt_actual INTO ls_actual WHERE fistl EQ ls_proj-wbsid. "#EC CI_NESTED
    ls_data-calendarmonth =  ls_actual-gjahr && ls_actual-perio+1.
    READ TABLE lt_fipex INTO ls_fipex WITH KEY fipex = ls_actual-fipex.
    IF sy-subrc = 0.
      ls_data-resourcetypeid = ls_fipex-res_type.
    ENDIF.
    ls_data-actualcost = ls_actual-dmbtr.
    IF ls_data-calendarmonth < sy-datum(6)." <= ls_data-forecastperiod.
      ls_data-actualcostfc = ls_actual-dmbtr.
    ENDIF.
    ls_data-actualquantity = ls_actual-zact_qty.
    ls_data-uom = ls_actual-zact_unit.
    ls_data-vendor = ls_actual-lifnr.
    READ TABLE lt_plan INTO ls_plan WITH KEY
              /cpd/fpid = ls_proj-wbselementobject
              0calmonth = ls_data-calendarmonth
              /cpd/frtyp = ls_data-resourcetypeid
              zcpm_vdor = ls_data-vendor
              .
    IF sy-subrc = 0.
      ls_data-resourceid = ls_plan-/cpd/fres.
      ls_data-costcenter = ls_plan-/cpd/cstcnt.
    ENDIF.
    COLLECT ls_data INTO lt_data.
    CLEAR:
      ls_data-calendarmonth,
      ls_data-resourcetypeid,
      ls_data-uom,
      ls_data-actualquantity,
      ls_data-actualcost,
      ls_data-actualcostfc,
      ls_data-vendor,
      ls_data-resourceid,
      ls_data-costcenter.
  ENDLOOP.
ENDIF.
READ TABLE lt_commit TRANSPORTING NO FIELDS WITH KEY fistl = ls_proj-wbsid.
IF sy-subrc = 0.
  lv_tabix = sy-tabix.
  "Commitment
  LOOP AT lt_commit INTO ls_commit WHERE fistl EQ ls_proj-wbsid. "#EC CI_NESTED
    ls_data-calendarmonth = ls_commit-gjahr && ls_commit-perio+1.

    READ TABLE lt_fipex INTO ls_fipex WITH KEY fipex = ls_commit-fipex.
    IF sy-subrc = 0.
      ls_data-resourcetypeid = ls_fipex-res_type.
    ENDIF.
    ls_data-vendor = ls_actual-lifnr.
    READ TABLE lt_plan INTO ls_plan WITH KEY
              /cpd/fpid = ls_proj-wbselementobject
              0calmonth = ls_data-calendarmonth
              /cpd/frtyp = ls_data-resourcetypeid
```

```
                zcpm_vdor = ls_data-vendor
                .
          IF sy-subrc = 0.
            ls_data-resourceid = ls_plan-/cpd/fres.
            ls_data-costcenter = ls_plan-/cpd/cstcnt.
          ENDIF.
          ls_data-commitment = ls_commit-fkbtr.

          COLLECT ls_data INTO lt_data.
          CLEAR:
           ls_data-calendarmonth,
           ls_data-resourcetypeid,
           ls_data-uom,
           ls_data-commitment,
           ls_data-resourceid,
           ls_data-costcenter.
        ENDLOOP.
      ENDIF.
      READ TABLE lt_budget_rel TRANSPORTING NO FIELDS WITH KEY fundsctr = ls_proj-wbsid.
      IF sy-subrc = 0.
        lv_tabix = sy-tabix.
        "Released Budget
        LOOP AT lt_budget_rel INTO ls_budget WHERE fundsctr EQ ls_proj-wbsid."AND budtype EQ
lc_budtype. "#EC CI_NESTED
          ls_data-calendarmonth = ls_budget-fiscyear && ls_budget-rpmax+1.
          READ TABLE lt_fipex INTO ls_fipex WITH KEY fipex = ls_budget-cmmtitem.
          IF sy-subrc = 0 .
            ls_data-resourcetypeid = ls_fipex-res_type.
          ENDIF.
          ls_data-releasedbudget =  ls_budget-lvalx.

          COLLECT ls_data INTO lt_data.

          CLEAR:
          ls_data-calendarmonth,
          ls_data-resourcetypeid,
          ls_data-uom,
          ls_data-releasedbudget.
        ENDLOOP.
      ENDIF.
      READ TABLE lt_budget_tra TRANSPORTING NO FIELDS WITH KEY fundsctr = ls_proj-wbsid.
      IF sy-subrc = 0.
        lv_tabix = sy-tabix.
        "Transferred Budget
        LOOP AT lt_budget_tra INTO ls_budget WHERE fundsctr EQ ls_proj-wbsid. "#EC CI_NESTED
          ls_data-calendarmonth = ls_budget-fiscyear && ls_budget-rpmax+1.
          READ TABLE lt_fipex INTO ls_fipex WITH KEY fipex = ls_budget-cmmtitem.
          IF sy-subrc = 0 .
            ls_data-resourcetypeid = ls_fipex-res_type.
          ENDIF.
          ls_data-transferredbudget =  ls_budget-lvalx.

          COLLECT ls_data INTO lt_data.

          CLEAR:
          ls_data-calendarmonth,
          ls_data-resourcetypeid,
```

```
        ls_data-uom,
        ls_data-transferredbudget.
       ENDLOOP.
     ENDIF.
     READ TABLE lt_budget_tar TRANSPORTING NO FIELDS WITH KEY fictr = ls_proj-wbsid.
     IF sy-subrc = 0.
      lv_tabix = sy-tabix.
      "Target Budget
      LOOP AT lt_budget_tar INTO DATA(ls_budget_tar) WHERE fictr EQ ls_proj-wbsid. "#EC
CI_NESTED
        ls_data-calendarmonth = ls_budget_tar-gjahr && '01'.
        READ TABLE lt_fipex INTO ls_fipex WITH KEY fipex = ls_budget_tar-fipex.
        IF sy-subrc = 0 .
          ls_data-resourcetypeid = ls_fipex-res_type.
        ENDIF.
        ls_data-targetbudget =  ls_budget_tar-ztarget.

        COLLECT ls_data INTO lt_data.

        CLEAR:
        ls_data-calendarmonth,
        ls_data-resourcetypeid,
        ls_data-uom,
        ls_data-targetbudget.
       ENDLOOP.
     ENDIF.
     READ TABLE lt_data TRANSPORTING NO FIELDS WITH KEY projectid = ls_data-projectid.
     IF sy-subrc <> 0.
      APPEND ls_data TO lt_data.
     ENDIF.

     CLEAR ls_data.
   ENDLOOP.
   IF lt_data[] IS NOT INITIAL.
*    Get Cost Center Description based on workbook cost center
     lr_kostl  = VALUE #( FOR wa_data IN lt_data ( sign = 'I' option = 'EQ' low = wa_data-costcenter ) ).
     SELECT DISTINCT cc~costcenter,cc~costcentername FROM i_costcentervh AS cc
      INTO TABLE @DATA(lt_costcenter)
      WHERE costcenter IN @lr_kostl.
*    Get Vendor Description based on workbook vendor
     lr_lifnr  = VALUE #( FOR wa_data IN lt_data ( sign = 'I' option = 'EQ' low = wa_data-vendor ) ).
     SELECT DISTINCT vnd~supplier,vnd~suppliername FROM zcon_cpm_ddl_i_supplier  AS vnd
      INTO TABLE @DATA(lt_supplier)
      WHERE supplier IN @lr_lifnr.
   ENDIF.
*   Data Formatting
   LOOP AT lt_data INTO ls_data.
     READ TABLE lt_user INTO DATA(ls_user) WITH KEY createdby = ls_data-createdby.
     IF sy-subrc = 0.
      ls_data-createdby = ls_user-username.
     ENDIF.
*    Cost center Description
     READ TABLE lt_costcenter INTO DATA(ls_costcenter) WITH KEY costcenter = ls_data-costcenter.
     IF sy-subrc = 0.
      ls_data-costcenter = ls_costcenter-costcentername.
     ENDIF.
*    Resource type Description
```

```
    READ TABLE lt_supplier INTO DATA(ls_supplier) WITH KEY supplier = ls_data-vendor.
    IF sy-subrc = 0.
      ls_data-vendor = ls_supplier-suppliername.
    ENDIF.
*   If UoM is empty, default to 'H'
    IF ls_data-uom IS INITIAL.
      ls_data-uom = lc_uom.
    ENDIF.
*   Resource type Description
    READ TABLE lt_restyp INTO DATA(ls_restyp) WITH KEY res_type_id = ls_data-resourcetypeid.
    IF sy-subrc = 0.
      ls_data-resourcetype = ls_restyp-description.
    ENDIF.
*   UoM Description
    READ TABLE lt_uomt INTO DATA(ls_uomt) WITH KEY uom = ls_data-uom.
    IF sy-subrc = 0.
      ls_data-uom = ls_uomt-text.
    ENDIF.
*   Resource Description
    READ TABLE lt_res INTO DATA(ls_res) WITH KEY resource_id = ls_data-resourceid.
    IF sy-subrc = 0.
      ls_data-resourcename = ls_res-description.
    ENDIF.
*   Project status Description
    READ TABLE lt_projstage INTO DATA(ls_projstage) WITH KEY stage = ls_data-projectstatus.
    IF sy-subrc = 0.
      ls_data-projectstatus = ls_projstage-description.
    ENDIF.
    "consumed
    ls_data-consumed = ls_data-actualcost + ls_data-commitment.
    "available
    ls_data-available = ls_data-releasedbudget - ls_data-consumed.
    " forecast eac
    ls_data-eaccost = ls_data-actualcostfc + ls_data-etccost.
    " forecast period
    READ TABLE lt_month_names INTO ls_month_names WITH KEY mnr = ls_data-forecastperiod+
4(2).
    IF sy-subrc = 0.
      CONCATENATE ls_month_names-ktx '-' ls_data-forecastperiod(4) INTO ls_data-forecastperiod.
    ENDIF.
    " forecast status
    READ TABLE lt_forecast_status_desc INTO DATA(ls_forecast_status_desc) WITH KEY status =
ls_data-forecaststatus.
    IF sy-subrc = 0.
      ls_data-forecaststatus = ls_forecast_status_desc-text.
    ENDIF.
    IF lv_reportingtype EQ lc_reportingtype_proj.
      CLEAR:
         ls_data-wbsid,
         ls_data-wbsname,
         ls_data-respcostcenter,
         ls_data-resourcetypeid,
         ls_data-resourcetype,
         ls_data-resourceid,
         ls_data-resourcename,
         ls_data-costcenter,
         ls_data-vendor.
```

```
    ELSEIF lv_reportingtype EQ lc_reportingtype_prps.
*     IF ls_data-wbsid IS INITIAL.
*       CONTINUE.
*     ENDIF.
      CLEAR:
         ls_data-resourcetypeid,
         ls_data-resourcetype,
         ls_data-resourceid,
         ls_data-resourcename,
         ls_data-costcenter,
         ls_data-vendor.
    ELSEIF lv_reportingtype EQ lc_reportingtype_ftyp.
*     IF ls_data-resourcetypeid IS INITIAL.
*       CONTINUE.
*     ENDIF.
      CLEAR:
       ls_data-resourceid,
       ls_data-resourcename,
       ls_data-costcenter,
       ls_data-vendor.
    ELSEIF lv_reportingtype EQ lc_reportingtype_fres.
      CLEAR:
       ls_data-forecastperiod,
       ls_data-forecaststatus,
       ls_data-actualcostfc,
       ls_data-etccost,
       ls_data-eaccost,
       ls_data-forecaststatus,
       ls_data-forecaststatus,
       ls_data-transferredbudget,
       ls_data-targetbudget,
       ls_data-releasedbudget,
       ls_data-available .
    ENDIF.

    IF lv_reportingview EQ lc_reportingview_overview.
      CLEAR:
        ls_data-calendarmonth.
    ELSEIF ls_data-calendarmonth IS NOT INITIAL.
      IF ls_data-calendarmonth NOT BETWEEN lv_period_from AND lv_period_to.
        CONTINUE.
      ENDIF.
*     Data formatting
      CALL FUNCTION 'CONVERSION_EXIT_PERI6_OUTPUT'
        EXPORTING
          input  = ls_data-calendarmonth
        IMPORTING
          output = ls_data-calendarmonth.

    ENDIF.

    MOVE-CORRESPONDING ls_data TO ls_result.
*   Summarize the numeric values based on the Key
    COLLECT ls_result INTO et_result.
  ENDLOOP.
```

```
ENDMETHOD.
```

```
IF sy-uname eq  'DTSRB2T' and io_event->mv_event_id EQ lc_event_repo_action.
 DATA: lr_service_manager TYPE REF TO  /bobf/if_tra_service_manager,
       lr_tran_manager    TYPE REF TO /bobf/if_tra_transaction_mgr,
       lt_mod             TYPE        /bobf/t_frw_modification,
       ls_mod             TYPE        /bobf/s_frw_modification,
       lo_change          TYPE REF TO /bobf/if_tra_change,
       lo_message         TYPE REF TO /bobf/if_frw_message,
       lt_messages1       TYPE /bobf/t_frw_message_k,
       lt_message         TYPE /bobf/cm_frw=>tt_frw,
       lt_change_attr     TYPE /bobf/t_frw_name,
       lr_hd_data         TYPE REF TO /cpd/s_mp_hdr_k.
 FIELD-SYMBOLS: <struct> TYPE any,
                <field>  TYPE any.
 CREATE DATA lr_hd_data.
 ASSIGN lr_hd_data->* TO <struct>.
 MOVE-CORRESPONDING cs_data TO <struct>.
 APPEND lc_name_sub_planfield TO lt_change_attr.
 APPEND lc_name_tech_center TO lt_change_attr.
 APPEND lc_name_sub_system TO lt_change_attr.

 ls_mod-node       = /cpd/if_mp_pws_bo_c=>sc_node-mp_hdr.
 ls_mod-change_mode = /bobf/if_frw_c=>sc_modify_update.
 ls_mod-key        = lr_hd_data->key.
 ls_mod-source_node = /cpd/if_mp_pws_bo_c=>sc_node-mp_hdr.
 ls_mod-source_key  = lr_hd_data->parent_key.
 ls_mod-data       = lr_hd_data.
 ls_mod-changed_fields       = lt_change_attr.
 APPEND ls_mod TO lt_mod.
 TRY.
    lr_service_manager = /bobf/cl_tra_serv_mgr_factory=>get_service_manager(
      iv_bo_key = /cpd/if_mp_pws_bo_c=>sc_bo_key ).
    CALL METHOD lr_service_manager->modify
      EXPORTING
        it_modification = lt_mod
      IMPORTING
        eo_change      = lo_change
        eo_message     = lo_message.

    CALL METHOD lo_message->get
      IMPORTING
        et_message = lt_message.
 ENDTRY.
ENDIF.
```

```
/BOBF/IF_FRW_C=>SC_MODIFY_PROPERTY_NODE_ATTR
DATA lo_change TYPE REF TO /bobf/if_frw_change.
lo_change = /bobf/cl_frw_factory=>get_change( ).
lo_change->get_changes(
    EXPORTING
      iv_change_mode = /bobf/if_frw_c=>sc_modify_load
    IMPORTING
      et_change      = DATA(lt_change)
  ).



CALL METHOD /cpd/cl_pws_ws_configuration=>get_rep_attr_details
          EXPORTING
            iv_group_profile_id = ls_mp_details-grprf_id
          IMPORTING
            et_attributes      = lt_attr.
```

```
*&---------------------------------------------------------------------*
*& Report ZDE_CON_CPMR_CPM_UPD_ZZFIELDS
*&---------------------------------------------------------------------*
*&
*&---------------------------------------------------------------------*
REPORT zde_con_cpmr_cpm_upd_zzfields.

SELECT * FROM /cpd/d_mp_hdr INTO TABLE @DATA(lt_mp).
SELECT * FROM /CPD/D_pfp_ph INTO TABLE @DATA(lt_ph).
SELECT * FROM /CPD/D_pfp_pv INTO TABLE @DATA(lt_pv).
LOOP AT lt_mp ASSIGNING FIELD-SYMBOL(<ls_mp>).
  clear <ls_mp>-zzext_cooper.
  clear <ls_mp>-zzcooper_detail.
  clear <ls_mp>-zzdev_status.
  clear <ls_mp>-zzboard_name.
  clear <ls_mp>-zzboard_date.
  clear <ls_mp>-zzcustomer_vis.
  clear <ls_mp>-zzcontract_typ.
  clear <ls_mp>-zzplan_field.
  clear <ls_mp>-zzsub_planfield.
  clear <ls_mp>-zztech_center.
  clear <ls_mp>-zzsub_system.
ENDLOOP.

LOOP AT lt_ph ASSIGNING FIELD-SYMBOL(<ls_ph>).
  clear <ls_ph>-zzplan_option.
  clear <ls_ph>-zzact_from.
  clear <ls_ph>-zzact_to.
ENDLOOP.
LOOP AT lt_pv ASSIGNING FIELD-SYMBOL(<ls_pv>).
  clear <ls_pv>-zzcomment.
ENDLOOP.
IF lt_mp[] IS NOT INITIAL.
  MODIFY /cpd/d_mp_hdr FROM TABLE lt_mp.
ENDIF.
IF lt_ph[] IS NOT INITIAL.
  MODIFY /cpd/d_pfp_ph FROM TABLE lt_ph.
ENDIF.
IF lt_pv[] IS NOT INITIAL.
  MODIFY /cpd/d_pfp_pv FROM TABLE lt_pv.
ENDIF.


 zde_cl_con_cpm_common_util=>get_instance( )->set_user_fields(
        EXPORTING
          iv_profidproj = ls_project_detail-project_profile
        importing
          es_user_fields = ls_user_fields
       ).
       MOVE-CORRESPONDING ls_user_fields to ls_wbs_element.
```

```
ENHANCEMENT 1  ZCON_CPMIE_PFP_K26_PRICING.    "active version
*---------------------------------------------------------------------------------------------------------------
*  Author                : Baobao Gao – DTSRB2T
*  Functional Owner      : Yanming Xu - AHOR2ME
*  Date                  : 15/10/2024
*  SAP Charm # / JIRA ID #  : CPMP-195
*  Description           : rate = AT Rate + DP Rate
*  RICEFW ID             : CPM021
*---------------------------------------------------------------------------------------------------------------
  CALL METHOD zcl_con_cpm_ie_pfp_plan_rate=>enhance_plan_rate
    EXPORTING
      iv_date  = lv_date
      iv_kostl = ls_cssl-kostl
      iv_kokrs = ls_cssl-kokrs
      iv_periv = lv_fy_variant
    IMPORTING
      ev_rate  = ev_amount
      .
ENDENHANCEMENT.
```

```
*&---------------------------------------------------------------------*
*& Report ZDE_PSR_UPDATE_WBS_CUST_FIELDS
*&---------------------------------------------------------------------*
*&
*&---------------------------------------------------------------------*
*---------------------------------------------------------------------------------------------------------------
*  Author               : Baobao Gao – DTSRB2T
*  Functional Owner     : Yanming Xu - AHOR2ME
*  Date                 : 20/08/2024
*  SAP Charm # / JIRA ID #  : CPMP-56
*  Description          : Map the user fields of the wbs elements to custom fields
*  RICEFW ID            : CPM-017
*---------------------------------------------------------------------------------------------------------------
REPORT zde_psr_update_wbs_cust_fields.
DATA:
 gv_posid    TYPE ps_posid.
SELECTION-SCREEN BEGIN OF BLOCK bl01 WITH FRAME TITLE TEXT-b01.
 PARAMETERS: r_wbs RADIOBUTTON GROUP g1 DEFAULT 'X',
          r_zz  RADIOBUTTON GROUP g1.
 SELECT-OPTIONS s_posid FOR gv_posid.
SELECTION-SCREEN END OF BLOCK bl01.

START-OF-SELECTION.
 IF r_zz EQ  abap_true.
   PERFORM frm_move_data.
 ELSEIF r_wbs EQ  abap_true.
*    Update data to prps
   PERFORM frm_process_data.
 ENDIF.
*&---------------------------------------------------------------------*
*& Form frm_process_data
*&---------------------------------------------------------------------*
*& text
*&---------------------------------------------------------------------*
*& -->  p1       text
*& <--  p2       text
*&---------------------------------------------------------------------*
FORM frm_process_data .
 DATA: lv_upd_err_flg TYPE char1
      ,lv_text_dev_status TYPE val_text
      ,lv_text_board_name TYPE val_text
      ,lv_text_customer_vis TYPE val_text
      ,lv_text_contract_typ TYPE val_text
      ,lv_text_external_cooperation TYPE zcpm_name60
      ,lv_text_cooperation_detail TYPE zcpm_name60
      .
 CONSTANTS:lc_domain_devstatus   TYPE domname VALUE 'ZDEV_STATUS',
         lc_domain_boardname   TYPE domname VALUE 'ZBOARD_NAME',
         lc_domain_customervis TYPE domname VALUE 'ZCUSTOMER_VIS',
         lc_domain_contracttyp TYPE domname VALUE 'ZCONTRACT_TYP'.
 " External Cooperation
 SELECT DISTINCT
   ExternalCooperation AS value,
   ExternalCooperationName AS  text
 FROM zcon_cpm_ddl_i_extcpn
```

```
  INTO TABLE @DATA(lt_text_external_cooperation).

 " Cooperation Detail
 SELECT DISTINCT
   cooper_detail AS value,
   CooperationDetailName AS  text
 FROM zcon_cpm_ddl_extdet
  INTO TABLE @DATA(lt_text_cooperation_detail).


 " Development Status
 SELECT DISTINCT
  domainvalue AS value,
  domaintext AS  text
 FROM i_domainfixedvaluetext
 WHERE sapdatadictionarydomain = @lc_domain_devstatus
INTO TABLE @DATA(lt_text_dev_status).
 " Board Name
 SELECT DISTINCT
   domainvalue AS value,
   domaintext AS  text
 FROM i_domainfixedvaluetext
 WHERE  sapdatadictionarydomain = @lc_domain_boardname
 INTO TABLE @DATA(lt_text_board_name).
 " Customer Visibility
 SELECT DISTINCT
  domainvalue AS value,
  domaintext AS  text
 FROM i_domainfixedvaluetext
 WHERE sapdatadictionarydomain = @lc_domain_customervis
INTO TABLE @DATA(lt_text_customer_vis).
 " Contract Type
 SELECT DISTINCT
  domainvalue AS value,
  domaintext AS  text
 FROM i_domainfixedvaluetext
 WHERE sapdatadictionarydomain = @lc_domain_contracttyp
INTO TABLE @DATA(lt_text_contract_typ).
* Read table prps
 SELECT
    pspnr,
    posid,
    usr00,
    usr02,
    usr03,
    usr04,
    usr09,
    usr10,
    zzext_cooper,
    zzcooper_detail,
    zzdev_status,
    zzboard_name,
    zzboard_date,
    zzcustomer_vis,
    zzcontract_typ
  FROM prps
  INTO TABLE @DATA(lt_prps)
```

```
   WHERE posid IN @s_posid.
  IF sy-subrc = 0.
   LOOP AT lt_prps ASSIGNING FIELD-SYMBOL(<ls_prps>).
    CLEAR:
lv_upd_err_flg,lv_text_dev_status,lv_text_board_name,lv_text_customer_vis,lv_text_contract_typ,l
v_text_external_cooperation,lv_text_cooperation_detail.

    "Development Status            -            USR02   Directly mapping - exception: "tbd" - "n.a."
    IF <ls_prps>-usr02 IS NOT INITIAL.
     lv_text_dev_status = <ls_prps>-usr02.
     IF <ls_prps>-usr02 CS 'tbd'
      OR <ls_prps>-usr02 CS 'Tbd'
      OR <ls_prps>-usr02 CS 'Weiterentw'
      OR <ls_prps>-usr02 CS 'Adv devlpm'
      .
      lv_text_dev_status = 'Advanced Development'.
*       <ls_prps>-zzdev_status = 'n.a.'.
     ELSEIF  <ls_prps>-usr02 CS 'Erstentw'
      OR <ls_prps>-usr02 CS 'Prototype'.
      lv_text_dev_status = 'Initial Development'.
     ENDIF.
     READ TABLE lt_text_dev_status INTO DATA(ls_text_dev_status) WITH KEY text =
lv_text_dev_status.
     IF sy-subrc = 0.
      <ls_prps>-zzdev_status = ls_text_dev_status-value.
     ENDIF.
    ENDIF.
*    Board name -  USR03  Directly mapping - exception: "tbd" - "Pending Approval"; "Sonstiges" -
"Other"
    IF <ls_prps>-usr03 IS NOT INITIAL.
     lv_text_board_name = <ls_prps>-usr03.
     IF <ls_prps>-usr03 CS 'tbd'
      OR  <ls_prps>-usr03 CS 'Tbd'.
      lv_text_board_name = 'Pending Approval'.
     ELSEIF <ls_prps>-usr03 CS 'Sonstige'.
      lv_text_board_name = 'Others'.
     ENDIF.
     READ TABLE lt_text_board_name INTO DATA(ls_text_board_name) WITH KEY text =
lv_text_board_name.
     IF sy-subrc = 0.
      <ls_prps>-zzboard_name = ls_text_board_name-value.
     ENDIF.
    ENDIF.
    "Board date -           USR09
    <ls_prps>-zzboard_date = <ls_prps>-usr09.
    "Customer Visibility   - USR04  If Useful Life is "36" then "Visible"; If Useful Life is "60" then "Not
Visible"
    IF <ls_prps>-usr04 IS NOT INITIAL.
     IF <ls_prps>-usr04 LE '36'.
      lv_text_customer_vis = 'Visible'.
     ELSE."IF <ls_prps>-usr04 EQ '60'.
      lv_text_customer_vis = 'Not Visible'.
     ENDIF.
     READ TABLE lt_text_customer_vis INTO DATA(ls_text_customer_vis) WITH KEY text =
lv_text_customer_vis.
     IF sy-subrc = 0.
      <ls_prps>-zzcustomer_vis = ls_text_customer_vis-value.
```

```
      ENDIF.
    ENDIF.
    "Contract Type -  USR02 (for S projects)  Directly mapping - exception: "tbd" - "n.a."
    IF <ls_prps>-usr02 IS NOT INITIAL AND <ls_prps>-posid+0(1) EQ 'S'.
      lv_text_contract_typ = <ls_prps>-usr02.
      IF <ls_prps>-usr02 CS 'tbd'
        OR <ls_prps>-usr02 CS 'Tbd'
        OR <ls_prps>-usr02 CS 'Dienstver.'
        OR <ls_prps>-usr02 CS 'Serv contr.'
        .
        lv_text_contract_typ = 'Service-Contract'.
      ELSEIF <ls_prps>-usr02 CS 'Werkvertr.'
      OR <ls_prps>-usr02 CS 'Work contr'
      .
        lv_text_contract_typ = 'Work-Contract'.
      ENDIF.
      READ TABLE lt_text_contract_typ INTO DATA(ls_text_contract_typ) WITH KEY text =
lv_text_contract_typ.
      IF sy-subrc = 0.
        <ls_prps>-zzcontract_typ = ls_text_contract_typ-value.
      ENDIF.
    ENDIF.
    IF <ls_prps>-usr10 EQ abap_true.
      "If USR10 Subsidized Proj is "Yes",
      "then set External Cooperation to "Subsidized Project", set Cooperation Detail to "n.a."
      lv_text_external_cooperation = 'Subsidized Project'.
      lv_text_cooperation_detail = 'n.a.'.
    ELSE.
      "Otherwise
      IF <ls_prps>-usr00 CS 'Ja-FTE & Sachkosten'
         OR <ls_prps>-usr00 CS 'Yes–FTE&Material Cst'.
        "If USR00 PACE Project is "Ja-FTE & Sachkkosten" or "Yes-FTE&Material Cst",
        "then set External Cooperation to "Alliance", set Cooperation Detail to "[net]FTE&Material";
        lv_text_external_cooperation = 'Alliance'.
        lv_text_cooperation_detail = '[net]FTE&Material'.
      ELSEIF <ls_prps>-usr00 CS 'Ja-kein Netting'
        OR <ls_prps>-usr00 CS 'Opt.-kein Netting'
        OR <ls_prps>-usr00 CS 'Yes-No Netting'.
        "If USR00 PACE Project is "Ja-kein Netting" or "Nein" or "No" Or "Yes-No Netting" or "Opt.-kein
Netting",
        "then set External Cooperation to "Alliance", set Cooperation Detail to "[net]no"
        lv_text_external_cooperation = 'Alliance'.
        lv_text_cooperation_detail = '[net]no'.
      ELSEIF <ls_prps>-usr00 CS 'Ja-SachkostenNetting'
      OR <ls_prps>-usr00 CS 'Yes–Material Cst'.
        "If USR00 PACE Project is "Ja-SachkostenNetting",
        "then set External Cooperation to "Alliance", set Cooperation Detail to "[net]Material"
        lv_text_external_cooperation = 'Alliance'.
        lv_text_cooperation_detail = '[net]Material'.
      ELSEIF <ls_prps>-usr00 CS 'Tbd'
        OR <ls_prps>-usr00 CS 'Nein'
        OR <ls_prps>-usr00 CS 'No'.
        "If USR00 PACE Project is "Tbd",
        "then set External Cooperation to "None", set Cooperation Detail to "n.a."
        lv_text_external_cooperation = 'None'.
        lv_text_cooperation_detail = 'n.a.'.
      ENDIF.
```

```
    ENDIF.
    "set External Cooperation
    IF lv_text_external_cooperation IS NOT INITIAL.
      READ TABLE lt_text_external_cooperation INTO DATA(ls_text_external_cooperation) WITH KEY
text = lv_text_external_cooperation.
      IF sy-subrc = 0.
        <ls_prps>-zzext_cooper = ls_text_external_cooperation-value.
      ENDIF.
    ENDIF.
    ""set Cooperation Detail
    IF lv_text_cooperation_detail IS NOT INITIAL.
      READ TABLE lt_text_cooperation_detail INTO DATA(ls_text_cooperation_detail) WITH KEY text =
lv_text_cooperation_detail.
      IF sy-subrc = 0.
        <ls_prps>-zzcooper_detail = ls_text_cooperation_detail-value.
      ENDIF.
    ENDIF.

    UPDATE prps SET
       zzdev_status = @<ls_prps>-zzdev_status,
       zzboard_name = @<ls_prps>-zzboard_name,
       zzboard_date = @<ls_prps>-zzboard_date,
       zzcustomer_vis = @<ls_prps>-zzcustomer_vis,
       zzcontract_typ = @<ls_prps>-zzcontract_typ,
       zzext_cooper = @<ls_prps>-zzext_cooper,
       zzcooper_detail = @<ls_prps>-zzcooper_detail
       WHERE pspnr = @<ls_prps>-pspnr.
    IF sy-subrc <> 0.
      lv_upd_err_flg = abap_true.
      EXIT.
    ENDIF.
  ENDLOOP.
  IF lv_upd_err_flg EQ abap_true.
    MESSAGE e036(cj) .
  ELSE.
    MESSAGE s222(cj) .
  ENDIF.
 ENDIF.

ENDFORM.
*&---------------------------------------------------------------------*
*& Form frm_move_data
*&---------------------------------------------------------------------*
*& text
*&---------------------------------------------------------------------*
*& -->  p1       text
*& <--  p2       text
*&---------------------------------------------------------------------*
FORM frm_move_data .

 SELECT * FROM /cpd/d_mp_hdr INTO TABLE @DATA(lt_mp).
 SELECT * FROM /CPD/D_pfp_ph INTO TABLE @DATA(lt_ph).
 SELECT * FROM /CPD/D_pfp_pv INTO TABLE @DATA(lt_pv).
 SELECT * FROM zcon_cpmt_cpmlog INTO TABLE @DATA(lt_log).

 LOOP AT lt_mp ASSIGNING FIELD-SYMBOL(<ls_mp>).
  IF <ls_mp>-ext_cooper IS NOT INITIAL.<ls_mp>-zzext_cooper = <ls_mp>-ext_cooper. ENDIF.
```

```
   IF <ls_mp>-cooper_detail IS NOT INITIAL.<ls_mp>-zzcooper_detail = <ls_mp>-cooper_detail.
ENDIF.
   IF <ls_mp>-dev_status IS NOT INITIAL.<ls_mp>-zzdev_status = <ls_mp>-dev_status. ENDIF.
   IF <ls_mp>-board_name IS NOT INITIAL.<ls_mp>-zzboard_name = <ls_mp>-board_name. ENDIF.
   IF <ls_mp>-board_date IS NOT INITIAL.<ls_mp>-zzboard_date = <ls_mp>-board_date. ENDIF.
   IF <ls_mp>-customer_vis IS NOT INITIAL.<ls_mp>-zzcustomer_vis = <ls_mp>-customer_vis. ENDIF.
   IF <ls_mp>-contract_typ IS NOT INITIAL.<ls_mp>-zzcontract_typ = <ls_mp>-contract_typ. ENDIF.
   IF <ls_mp>-plan_field IS NOT INITIAL.<ls_mp>-zzplan_field = <ls_mp>-plan_field. ENDIF.
   IF <ls_mp>-sub_planfield IS NOT INITIAL.<ls_mp>-zzsub_planfield = <ls_mp>-sub_planfield. ENDIF.
   IF <ls_mp>-tech_center IS NOT INITIAL.<ls_mp>-zztech_center = <ls_mp>-tech_center. ENDIF.
   IF <ls_mp>-sub_system IS NOT INITIAL.<ls_mp>-zzsub_system = <ls_mp>-sub_system. ENDIF.
  ENDLOOP.

  LOOP AT lt_ph ASSIGNING FIELD-SYMBOL(<ls_ph>).
   IF <ls_ph>-plan_option IS NOT INITIAL.<ls_ph>-zzplan_option = <ls_ph>-plan_option. ENDIF.
   IF <ls_ph>-act_from IS NOT INITIAL.<ls_ph>-zzact_from = <ls_ph>-act_from. ENDIF.
   IF <ls_ph>-act_to IS NOT INITIAL.<ls_ph>-zzact_to = <ls_ph>-act_to. ENDIF.
  ENDLOOP.


  LOOP AT lt_pv ASSIGNING FIELD-SYMBOL(<ls_pv>).
   IF <ls_pv>-comment IS NOT INITIAL.<ls_pv>-zzcomment = <ls_pv>-comment. ENDIF.
  ENDLOOP.

  LOOP AT lt_log ASSIGNING FIELD-SYMBOL(<ls_log>).
   IF <ls_log>-ext_cooper IS NOT INITIAL.<ls_log>-zzext_cooper = <ls_log>-ext_cooper. ENDIF.
   IF <ls_log>-cooper_detail IS NOT INITIAL.<ls_log>-zzcooper_detail = <ls_log>-cooper_detail. ENDIF.
   IF <ls_log>-dev_status IS NOT INITIAL.<ls_log>-zzdev_status = <ls_log>-dev_status. ENDIF.
   IF <ls_log>-board_name IS NOT INITIAL.<ls_log>-zzboard_name = <ls_log>-board_name. ENDIF.
   IF <ls_log>-board_date IS NOT INITIAL.<ls_log>-zzboard_date = <ls_log>-board_date. ENDIF.
   IF <ls_log>-customer_vis IS NOT INITIAL.<ls_log>-zzcustomer_vis = <ls_log>-customer_vis. ENDIF.
   IF <ls_log>-contract_typ IS NOT INITIAL.<ls_log>-zzcontract_typ = <ls_log>-contract_typ. ENDIF.
   IF <ls_log>-plan_field IS NOT INITIAL.<ls_log>-zzplan_field = <ls_log>-plan_field. ENDIF.
   IF <ls_log>-sub_planfield IS NOT INITIAL.<ls_log>-zzsub_planfield = <ls_log>-sub_planfield. ENDIF.
   IF <ls_log>-tech_center IS NOT INITIAL.<ls_log>-zztech_center = <ls_log>-tech_center. ENDIF.
   IF <ls_log>-sub_system IS NOT INITIAL.<ls_log>-zzsub_system = <ls_log>-sub_system. ENDIF.
  ENDLOOP.

  IF lt_mp[] IS NOT INITIAL.
   MODIFY /cpd/d_mp_hdr FROM TABLE lt_mp.
  ENDIF.
  IF lt_ph[] IS NOT INITIAL.
   MODIFY /cpd/d_pfp_ph FROM TABLE lt_ph.
  ENDIF.
  IF lt_pv[] IS NOT INITIAL.
   MODIFY /cpd/d_pfp_pv FROM TABLE lt_pv.
  ENDIF.
  IF lt_log[] IS NOT INITIAL.
   MODIFY zcon_cpmt_cpmlog FROM TABLE lt_log.
  ENDIF.
ENDFORM.
```

```
*---------------------------------------------------------------------*
*       CLASS /CPD/CL_PFP_SC_WBS_COST_UPDATE DEFINITION
*---------------------------------------------------------------------*
*
*---------------------------------------------------------------------*
class ZCL_CON_CPM_PFP_WBS_COST_UPD definition
  public
  create public .

public section.

  interfaces /CPD/IF_PFP_DOWNPORT_METHOD .

  types:
    BEGIN OF ty_objid,
       objnr TYPE j_objnr,
     END OF ty_objid .
  types:
    BEGIN OF ty_dtrmn_vln_dt,
       plan_break_down   TYPE /cpd/pfp_period,
       lv_year(4)        TYPE   n,
       lv_period(3)      TYPE   c,
       lv_co_area        TYPE kokrs,
       lt_valuation_date TYPE /cpd/t_pfp_valution_dates,
     END OF ty_dtrmn_vln_dt .
  types:
    ty_t_objid TYPE SORTED TABLE OF ty_objid WITH UNIQUE KEY objnr .
  types:
    ty_t_dtrmn_vln_dt TYPE TABLE OF ty_dtrmn_vln_dt .

  class-data MV_STRUC type ref to DATA .
  PROTECTED SECTION.

    METHODS determine_valuation_date
      IMPORTING
        !ir_plan_data       TYPE REF TO data
        !is_plan_header     TYPE /cpd/s_pfp_plan_header_d
        !iv_co_area         TYPE kokrs
      EXPORTING
        !et_message         TYPE /cpd/t_message
        !et_valuation_dates TYPE /cpd/t_pfp_valution_dates .
  PRIVATE SECTION.

    METHODS delete_planned_data
      IMPORTING
        !it_hier_struct TYPE /cpd/t_sc_pfp_execution_hier
        !it_objid       TYPE ty_t_objid
        !iv_co_version  TYPE versn .
ENDCLASS.
```

```
CLASS ZCL_CON_CPM_PFP_WBS_COST_UPD IMPLEMENTATION.


* <SIGNATURE>---------------------------------------------------------------------------------------+
* | Instance Public Method
ZCL_CON_CPM_PFP_WBS_COST_UPD->/CPD/IF_PFP_DOWNPORT_METHOD~EXECUTE_DOWNPORT
* +---------------------------------------------------------------------------------------------+
* | [--->] IS_PLAN_HEADER              TYPE       /CPD/S_PFP_PLAN_HEADER_D(optional)
* | [--->] IR_DATA                   TYPE REF TO DATA
* | [--->] IV_CO_VERSION               TYPE       /CPD/PFP_MAP_CO_VER(optional)
* | [--->] IT_MAPPING_FIELDS           TYPE       /CPD/T_PFP_TRANSFER_MAPPING(optional)
* | [--->] IT_MAP_FIELDS               TYPE REF TO DATA
* | [--->] IV_SIMULATE_TRANSFER        TYPE       /CPD/PFP_FLAG_SIMULATE(optional)
* | [--->] IT_HIER_STRUCT              TYPE       /CPD/T_SC_PFP_EXECUTION_HIER
* | [--->] IT_VERSION                TYPE       /CPD/T_PFP_PLAN_VERSION(optional)
* | [--->] IV_VER_DEL_FLAG             TYPE       BOOLEAN(optional)
* | [--->] IS_ADDON_STRUCT             TYPE       /CPD/S_PFP_TRANS_ADDON(optional)
* | [<---] ET_MESSAGE                TYPE       /CPD/T_MESSAGE
* | [<---] ET_OP_DATA                TYPE REF TO DATA
* +---------------------------------------------------------------------------------------------</SIGNATURE>
  METHOD /cpd/if_pfp_downport_method~execute_downport.
** The method calls the relevant BAPI for cost update at WBS level.

   TYPES : BEGIN OF wbs_cost_detail,
           lv_wbs         TYPE  ps_posid,
           lv_cost_element TYPE  kstar,
           lv_act_type     TYPE  lstar,
           lv_cost_center  TYPE  kostl,
           lv_year(4)      TYPE   n,
           lv_month(2)     TYPE   c,
           lv_period(3)    TYPE   c,
           lv_amount       TYPE bapicurr_d,
           lv_qty         TYPE mefxxx,
           lv_curr        TYPE twaer,
           lv_uom          TYPE meinh,
          END OF wbs_cost_detail,

          BEGIN OF ty_res_fields,
            res_type  TYPE  /cpd/pfp_res_cl_name,
            lt_fields TYPE  /cpd/pfpt_sel_fields,
          END OF ty_res_fields,
          BEGIN OF ty_wbs_obj,
            object_id TYPE /cpd/object_id,
            wbs_id    TYPE ps_posid,
          END OF ty_wbs_obj.
   TYPES:
     BEGIN OF  ts_activity_type,
       lv_co_area  TYPE   kokrs,
       lv_activity TYPE   lstar,
       lv_act_unit TYPE leinh,
     END OF  ts_activity_type.
   DATA: lt_activity_type_dt TYPE SORTED TABLE OF ts_activity_type WITH UNIQUE KEY lv_co_area
lv_activity,
         ls_activity_type_dt TYPE           ts_activity_type,
         lt_det_val_date     TYPE ty_t_dtrmn_vln_dt,
         ls_det_val_date     TYPE ty_dtrmn_vln_dt.
```

```
****   Add buffer tables for -- Controlling area , activity type details , PS Project, netwrok activity
details
   TYPES : BEGIN OF ts_controlling_area,
           lv_co_area    TYPE  kokrs,
           lv_fy_variant TYPE  periv,
           lv_exrt       TYPE  /cpd/pfp_ex_rate_ty,
          END OF ts_controlling_area.

   TYPES : BEGIN OF wbs_cost_elem,
           lv_wbs         TYPE  ps_posid,
           lv_cost_element TYPE  kstar,
           lv_act_type     TYPE  lstar,
           lv_cost_center  TYPE  kostl,
           lv_co_area      TYPE  kokrs,
          END OF wbs_cost_elem.

*** Local fields for BAPI ' POST PRIMARY COST'.
   DATA : ls_mapping_field TYPE  /cpd/fc_trnmt_tf,
          ls_header_info   TYPE  bapiplnhdr,
          ls_indextab      TYPE  bapiacpstru,
          ls_coobj         TYPE  bapipcpobj,
          ls_pervalue      TYPE  bapipcpval,
          ls_message       TYPE  /cpd/s_message,
          ls_bapiret       TYPE  bapiret2,
          lt_message       TYPE  /cpd/t_message,
          ls_act_detail    TYPE  bapi1031_atoutputlist,
          lt_indextab      TYPE TABLE OF  bapiacpstru,
          lt_coobj         TYPE TABLE OF bapipcpobj,
          lt_pervalue      TYPE TABLE OF bapipcpval,
          lt_bapiret       TYPE TABLE OF bapiret2.

   DATA   ls_co_area_detail       TYPE bapi0004_2.
   DATA   ls_coabapiret           TYPE bapireturn.


   DATA: ls_restyp_cost_element  TYPE        /cpd/fc_rty_cst,
         ls_res_cost_element     TYPE        /cpd/fc_res_cst,
         ls_default_cost_element TYPE        /cpd/fc_kf_ceat,
         lt_restyp_cost_element  TYPE TABLE OF /cpd/fc_rty_cst,
         lt_res_cost_element     TYPE TABLE OF /cpd/fc_res_cst.

   DATA : ls_act_indextab       TYPE    bapiacistru,
          ls_act_coobj          TYPE    bapiaciobj,
          ls_act_pervalue       TYPE    bapiacival,
          lt_act_indextab       TYPE TABLE OF  bapiacistru,
          lt_act_coobj          TYPE TABLE OF  bapiaciobj,
          lt_act_pervalue       TYPE TABLE OF  bapiacival,
          lt_res_fields         TYPE STANDARD TABLE OF ty_res_fields,
          ls_res_fields         TYPE ty_res_fields,
          lt_controlling_area_dt TYPE  TABLE OF   ts_controlling_area,
          ls_controlling_area_dt TYPE           ts_controlling_area.


   DATA : lv_fiscal_year(4)    TYPE     n,
          lv_count             TYPE     i,
          lv_period_from       TYPE     co_perab,
          lv_period_to         TYPE     co_perab,
```

```
        lv_period          TYPE     string,
        lv_key_figure      TYPE     /cpd/pfp_key_figure,
        lv_struct_field(11)  VALUE    'VAR_VAL_PER',
        lv_quant_field(16)   VALUE    'QUANTITY_VAR_PER',
        lv_plan_curr       TYPE     twaer,
*         lv_sender_cost_center TYPE     kostl,
        lv_co_area         TYPE     kokrs,
        lv_imple_class     TYPE     /cpd/pfp_res_cl_name,
        lv_calmonth        TYPE     char7,
        ls_wbs_cost_detail   TYPE     wbs_cost_detail,
        ls_first_wbs_rec    TYPE     wbs_cost_detail,
        ls_wbs_cost_elem    TYPE     wbs_cost_elem,
        lv_wbs_id          TYPE     ps_posid,
        lv_exrt           TYPE     /cpd/pfp_ex_rate_ty,
        lv_c_amt           TYPE     bapicurr_d,
        lv_ce_amt          TYPE     wkgxxx,
        lv_co_flag         TYPE     c,
        lv_fy_variant      TYPE     periv,
        lv_date            TYPE     sy-datum,
        lv_posting_period(3) TYPE     n,
        lv_posting_year(4)   TYPE     n,
        lv_posting_pd(2)    TYPE     c,
        lv_curr_method      TYPE     string,
        lv_trans_met_id     TYPE     string,
        lv_wbs_s           TYPE     string,
        lv_curr_method_s    TYPE     string,
        lv_resource_error   TYPE     boolean,
        ls_temp_wbs        TYPE     wbs_cost_detail,
        lt_wbs_cost_detail   TYPE TABLE OF wbs_cost_detail,
        lt_wbs_cost_elem    TYPE SORTED TABLE OF wbs_cost_elem WITH UNIQUE KEY lv_wbs
lv_cost_element lv_act_type lv_cost_center,
        lo_pfp_erp         TYPE REF TO /cpd/cl_pfp_erp,
        lt_hier_struct      TYPE /cpd/t_pfp_execution_hier,
        ls_hier_struct      TYPE /cpd/s_pfp_execution_hier,
        lt_hier_struct_tmp   TYPE /cpd/t_pfp_execution_hier,
        lt_wbs_obj         TYPE SORTED TABLE OF ty_wbs_obj WITH UNIQUE KEY object_id,
        ls_wbs_obj         TYPE ty_wbs_obj,
        lv_cproj_flag       TYPE boolean,
        lt_objid           TYPE  ty_t_objid,
        ls_objid           TYPE ty_objid,
        lv_valuation_date    TYPE ck_steas.

    DATA: lr_if_pfp_resource TYPE REF TO     /cpd/if_pfp_resource,
        lr_error          TYPE REF TO     cx_dynamic_check,
        lr_badi_input      TYPE REF TO     /cpd/pfp_badi_erp_transfer,
        lr_data           TYPE REF TO     data,
        lt_fields         TYPE          /cpd/pfpt_sel_fields,
        lv_search_help     TYPE          shlpname,
        lv_mstr(200)       TYPE          c,
        lv_key_figure_cost TYPE          /cpd/pfp_key_figure,
        lv_key_figure_rev  TYPE          /cpd/pfp_key_figure,
        lv_implement_class TYPE          /cpd/pfp_implement_class,
        lv_co_version      TYPE          /cpd/pfp_map_co_ver,
        lv_plan_breakdown  TYPE          /cpd/pfp_period,
        lr_plan_data       TYPE REF TO data,
        lt_valuation_date  TYPE /cpd/t_pfp_valution_dates,
        ls_valuation_date  TYPE /cpd/s_pfp_valution_dates,
```

```
      ltab_message      TYPE /cpd/t_message,
      lv_amount         TYPE ck_price_total,
      lv_qty            TYPE mefxxx,
      lt_fc_rty         TYPE TABLE OF /cpd/fc_rty,
      ls_fc_rty         TYPE /cpd/fc_rty.

  FIELD-SYMBOLS :
   <fs_plan_data>   TYPE ANY TABLE,
   <fs_line_item>   TYPE any,
   <fs_co_area>     TYPE any,
   <fs_data>        TYPE any,
   <fv_co_ver>      TYPE any,
   <fs_cost_center> TYPE any,
   <fs_restype>     TYPE any,
   <fs_calmonth>    TYPE any,
   <fs_qty>         TYPE any,
   <fs_period>      TYPE any,
   <fs_wbs_fpid>    TYPE any,
   <fs_curr>        TYPE any,
   <fs_res>         TYPE any,
   <fs_uom>         TYPE any,
   <fs_value>       TYPE any.

  DATA: lt_wbs TYPE TABLE OF  /cpd/s_pfp_wbs_co,
      ls_wbs TYPE          /cpd/s_pfp_wbs_co.
  CONSTANTS: lc_act_type_2000 TYPE  lstar VALUE '2000'.

** For Bid structure skip the trasfer and give a message
  DATA: ls_mapping_fields TYPE /cpd/fc_trnmt_tf.

*Fetching method id into a local variable
  IF it_mapping_fields IS NOT INITIAL.
   READ TABLE it_mapping_fields INTO ls_mapping_fields INDEX 1.
   lv_trans_met_id = ls_mapping_fields-trans_met_id.
  ENDIF.

  IF it_hier_struct IS INITIAL.

   IF sy-subrc EQ 0.
    ls_message-msgty     = 'E'.
    ls_message-msgid     = /cpd/cl_pfp_constants=>gc_msg_class_name. "'/CPD/PFP_MESSAGES'.
    ls_message-msgno     = '262'.
    ls_message-msgv1     = lv_trans_met_id.
    APPEND ls_message TO et_message.
   ENDIF.

  ELSE.

*  Getting all Resource Type Master Data
   SELECT * FROM /cpd/fc_rty INTO TABLE lt_fc_rty.
   SORT lt_fc_rty BY res_type_id.

   lt_hier_struct[] = it_hier_struct.
   SORT lt_hier_struct BY object_id.
   lt_hier_struct_tmp[] =  lt_hier_struct[].
   READ TABLE it_hier_struct TRANSPORTING NO FIELDS WITH KEY object_link = '0DPO'.
   IF sy-subrc = 0.
```

```
      lv_cproj_flag = abap_true.
    ENDIF.

    "Handle Multiple Co Version
    lv_co_version = iv_co_version.

    ASSIGN it_map_fields->* TO <fs_plan_data>.
    IF <fs_plan_data> IS ASSIGNED.
      LOOP AT <fs_plan_data> ASSIGNING <fs_data>.
        EXIT.
      ENDLOOP.

      IF <fs_data> IS ASSIGNED.
        ASSIGN COMPONENT 'VER' OF STRUCTURE <fs_data> TO <fv_co_ver>.
        IF <fv_co_ver> IS ASSIGNED AND <fv_co_ver> NE ''.
          lv_co_version = <fv_co_ver>.
        ENDIF.
      ENDIF.
    ENDIF.


    lv_plan_breakdown    = /cpd/cl_pfp_erp=>mv_period.

    IF lv_co_version IS NOT INITIAL.
**** 1. Read customizing for mapping fields. strucutre "ls_wbs_cust_data" will contain the
customizing values.
      READ TABLE it_mapping_fields INTO ls_mapping_field WITH KEY trgt_fld_name =
'KEY_FIG_COST'.
      IF sy-subrc = 0.
        lv_key_figure_cost =   ls_mapping_field-field_value.
      ENDIF.

      IF ( lv_key_figure_cost IS INITIAL ). " Throw error if cost key figure is not maintaied.
        ls_message-msgty     = 'E'.
        ls_message-msgid     = '/CPD/PFP_MESSAGES'.
        ls_message-msgno     = '246'.
        APPEND ls_message TO et_message.

      ELSE.
*** Assign the key figure element.
        IF lv_key_figure_cost IS NOT INITIAL.
          lv_key_figure = lv_key_figure_cost.
          lv_curr_method  = TEXT-001.
        ENDIF.
*        IF iv_ver_del_flag = abap_false.
****2. Read the Planning data.
        IF <fs_plan_data> IS ASSIGNED.
          TRY.
              GET BADI lr_badi_input.
            CATCH cx_badi_not_implemented.
          ENDTRY.

          IF lr_badi_input IS NOT INITIAL.
            SELECT SINGLE implement_class FROM /cpd/fc_trnmt INTO lv_implement_class WHERE
trans_met_id = lv_trans_met_id.
          ENDIF.
```

```
        LOOP AT <fs_plan_data> ASSIGNING <fs_data>. "#EC LOOP_ASSIG
*** Call BADI to change input mapping parameteres
          IF lr_badi_input IS NOT INITIAL AND lv_implement_class IS NOT INITIAL.
            GET REFERENCE OF <fs_data> INTO lr_data.
            CALL BADI lr_badi_input->modify_input_parameters
              EXPORTING
                iv_implementing_class = lv_implement_class
                it_hier_struct       = it_hier_struct
                is_plan_header       = is_plan_header
              CHANGING
                cs_data             = lr_data.

            ASSIGN lr_data->* TO <fs_data>.
          ENDIF.

          IF <fs_data> IS ASSIGNED.
            CLEAR ls_wbs_cost_detail.

            IF lv_key_figure_cost IS NOT INITIAL.                          " Note 3150380
              ASSIGN COMPONENT 'KEY_FIG_COST' OF STRUCTURE <fs_data> TO <fs_value>.
              IF <fs_value> IS ASSIGNED AND <fs_value> IS INITIAL.
                CONTINUE.
              ENDIF.
            ENDIF.

*             IF sy-tabix = 1 OR lv_co_area IS INITIAL. " Note 2484994
*** Read The controlling area.
            ASSIGN COMPONENT 'CO_AREA' OF STRUCTURE <fs_data> TO <fs_co_area>.
            IF <fs_co_area> IS ASSIGNED.
              lv_co_area =  <fs_co_area>.
              CLEAR ls_controlling_area_dt.
              READ TABLE lt_controlling_area_dt INTO ls_controlling_area_dt WITH KEY  lv_co_area =
lv_co_area.
              IF sy-subrc <> 0.
*  For currency conversion, get the co area currency.
                CALL FUNCTION 'BAPI_CONTROLLINGAREA_GETDETAIL'
                  EXPORTING
                    controllingareaid     = lv_co_area
                  IMPORTING
                    controllingarea_detail = ls_co_area_detail
                    return                = ls_coabapiret.
                IF ls_coabapiret IS INITIAL AND ls_co_area_detail IS NOT INITIAL.
                  lv_fy_variant = ls_co_area_detail-fy_variant.
                  ls_controlling_area_dt-lv_fy_variant = ls_co_area_detail-fy_variant.
                  ls_controlling_area_dt-lv_co_area    = lv_co_area.
                  SELECT SINGLE exchange_rate_type FROM /cpd/fc_pl_scen INTO lv_exrt WHERE
plan_scen_id = is_plan_header-plan_scen_id .
                  ls_controlling_area_dt-lv_exrt = lv_exrt.
                  APPEND ls_controlling_area_dt TO lt_controlling_area_dt.
                ENDIF.
              ELSE.
                lv_fy_variant = ls_controlling_area_dt-lv_fy_variant.
                lv_exrt      = ls_controlling_area_dt-lv_exrt.
              ENDIF. "sy-subrc <> 0
            ENDIF.
*             ENDIF. " Note 2484994
            IF lv_co_area IS INITIAL.
```

```
            lv_co_flag = 'X'.
           ELSE.
****3. Read WBS
            ASSIGN COMPONENT 'WBS_ELEMENT' OF STRUCTURE <fs_data> TO <fs_wbs_fpid>.
            IF <fs_wbs_fpid> IS ASSIGNED.
             IF lv_cproj_flag <> abap_true.
              READ TABLE lt_objid TRANSPORTING NO FIELDS WITH KEY objnr = <fs_wbs_fpid>
BINARY SEARCH.
               IF sy-subrc <> 0.
                ls_objid-objnr = <fs_wbs_fpid>.
                INSERT ls_objid INTO TABLE lt_objid.
                CLEAR ls_objid.
               ENDIF.
              ENDIF.
*** Get the WBS ID from the OBJNR
              READ TABLE lt_wbs_obj INTO ls_wbs_obj WITH KEY object_id = <fs_wbs_fpid> BINARY
SEARCH.
              IF sy-subrc = 0.
               ls_wbs_cost_detail-lv_wbs = ls_wbs_obj-wbs_id.
              ELSE.
               READ TABLE lt_hier_struct INTO ls_hier_struct WITH KEY object_id = <fs_wbs_fpid>
BINARY SEARCH.
               IF sy-subrc = 0.
                lv_wbs_id = ls_hier_struct-text.
                ls_wbs_cost_detail-lv_wbs  = lv_wbs_id.
                ls_wbs_obj-object_id = <fs_wbs_fpid>.
                ls_wbs_obj-wbs_id = lv_wbs_id.
                INSERT ls_wbs_obj INTO TABLE lt_wbs_obj.
                DELETE lt_hier_struct_tmp WHERE object_id = ls_hier_struct-object_id.
               ELSE.
                SELECT SINGLE posid FROM prps INTO lv_wbs_id WHERE objnr = <fs_wbs_fpid> . "#EC
CI_NOFIELD
                IF sy-subrc = 0.
                 ls_wbs_cost_detail-lv_wbs  = lv_wbs_id.
                 ls_wbs_obj-object_id = <fs_wbs_fpid>.
                 ls_wbs_obj-wbs_id = lv_wbs_id.
                 INSERT ls_wbs_obj INTO TABLE lt_wbs_obj.
                ENDIF.
               ENDIF.
              ENDIF.
***4. Get the Cost Element/ Activity type assigned to resource /Resource type. Read Rsource from
the line item
              CLEAR lv_resource_error.
              ASSIGN COMPONENT 'RTYPE' OF STRUCTURE <fs_data> TO <fs_restype>.
              IF <fs_restype> IS ASSIGNED.
               ASSIGN COMPONENT 'RES' OF STRUCTURE <fs_data> TO <fs_res>.
               IF <fs_res> IS ASSIGNED.
*                  IF  <fs_restype> IS NOT INITIAL AND <fs_res> IS INITIAL.  " Check if planning is done
on resource.
*                   lv_resource_error = 'X'.
*                   CLEAR ls_message.
*                   READ TABLE et_message WITH KEY msgid = /cpd/cl_pfp_constants=>
gc_msg_class_name   msgno = '410'  msgv1 = ls_wbs_cost_detail-lv_wbs  msgv2 = <fs_restype>
TRANSPORTING NO FIELDS.
*                   IF sy-subrc <> 0.
*                    ls_message-msgty     = /cpd/cl_pfp_constants=>gc_e.
*                    ls_message-msgid     = /cpd/cl_pfp_constants=>gc_msg_class_name.
```

```
*                    ls_message-msgno     = '410'.
*                    ls_message-msgv1     = ls_wbs_cost_detail-lv_wbs.
*                    ls_message-msgv2     = <fs_restype>.
*                    APPEND ls_message TO et_message.
*                   ENDIF.
*                  ELSE.
```

***4.2 Check resource type implementation class. IF it is of Internal activity type, resource is the activity type itself.

```
                CLEAR ls_fc_rty.
                READ TABLE lt_fc_rty INTO ls_fc_rty WITH KEY res_type_id = <fs_restype> BINARY
SEARCH.
                IF sy-subrc = 0.
                 lv_imple_class = ls_fc_rty-restype_class_name.
                ENDIF.
                CLEAR ls_res_fields.
                READ TABLE lt_res_fields INTO ls_res_fields WITH KEY res_type = <fs_restype>.
                IF sy-subrc <> 0.
                 TRY.
                    CREATE OBJECT lr_if_pfp_resource TYPE (lv_imple_class).
                    CALL METHOD lr_if_pfp_resource->get_search_help
                     IMPORTING
                       ex_search_help_name = lv_search_help.
                    IF sy-subrc EQ 0.
                     CLEAR lt_fields.
                     CALL METHOD /cpd/cl_pfp_po_services=>get_sh_fields_description
                       EXPORTING
                         iv_shlp_name = lv_search_help
                       IMPORTING
                         it_fields    = lt_fields.
                    ENDIF.
                   CATCH cx_sy_create_object_error INTO lr_error.
                    lv_mstr = lr_error->get_text( ).
                    ls_message-msgty  = 'W'.
                    ls_message-msgid = '/CPD/PFP_MESSAGES'.
                    ls_message-msgno = '245'.
                    ls_message-msgv1 = lv_mstr+0(50).
                    ls_message-msgv2 = lv_mstr+50(50).
                    ls_message-msgv3 = lv_mstr+100(50).
                    ls_message-msgv4 = lv_mstr+150(50).
                    APPEND ls_message TO et_message.
                 ENDTRY.
                 ls_res_fields-res_type = <fs_restype>.
                 APPEND LINES OF lt_fields TO ls_res_fields-lt_fields.
                 APPEND ls_res_fields TO lt_res_fields.
                ENDIF.
                READ TABLE ls_res_fields-lt_fields  TRANSPORTING NO FIELDS WITH KEY tabname =
'CSLA'.
                IF sy-subrc = 0 AND lv_key_figure_cost IS NOT INITIAL.
                 ls_wbs_cost_detail-lv_act_type = <fs_res>.
                ELSE.
*** 4.3 The resource typs isnot 'ACT TYPE' , Fetch the cost element from the customzing. Check Cost
element for that resource.
                 CLEAR : ls_restyp_cost_element , ls_res_cost_element.
                 READ TABLE lt_res_cost_element INTO ls_res_cost_element WITH KEY resource_id =
<fs_res> key_figure = lv_key_figure.
                 IF sy-subrc <> 0. " read the db table
```

```
                SELECT SINGLE * FROM /cpd/fc_res_cst INTO ls_res_cost_element WHERE
resource_id = <fs_res> AND key_figure = lv_key_figure.
                IF ( sy-subrc = 0 ).
                 APPEND ls_res_cost_element TO lt_res_cost_element.
                ELSE.      " means no cost element is maintained for Resource. Chk cost element for
Reource type then.
**** 4.3.1 Check Cost element for that resource Type.
                 READ TABLE lt_restyp_cost_element INTO ls_restyp_cost_element WITH KEY
res_type_id = <fs_restype>
                                                    plan_scen_id = is_plan_header-plan_scen_id
key_figure = lv_key_figure.
                 IF sy-subrc <> 0.
                  SELECT SINGLE * FROM /cpd/fc_rty_cst INTO  ls_restyp_cost_element WHERE
plan_scen_id = is_plan_header-plan_scen_id AND res_type_id = <fs_restype>
                                                      AND   key_figure = lv_key_figure.
                  IF sy-subrc = 0.
                   APPEND ls_restyp_cost_element TO lt_restyp_cost_element.
                  ENDIF.
                 ENDIF.
                ENDIF.   " IF ( sy-subrc = 0 ).
               ENDIF.   "  IF sy-subrc <> 0.

*** 4.3.2 Read Cost ELEMENT/Activity Type  for Key figure Cost.
               IF ls_res_cost_element IS NOT INITIAL.
                IF ls_res_cost_element-cost_element IS NOT INITIAL.
                 ls_wbs_cost_detail-lv_cost_element  = ls_res_cost_element-cost_element.
                ELSE.
                 ls_wbs_cost_detail-lv_act_type = ls_res_cost_element-activity_type.
                ENDIF.
               ELSE.
                IF ls_restyp_cost_element-cost_element IS NOT INITIAL.
                 ls_wbs_cost_detail-lv_cost_element  = ls_restyp_cost_element-cost_element.
                ELSE.
                 ls_wbs_cost_detail-lv_act_type = ls_restyp_cost_element-activity_type.
                ENDIF.
               ENDIF.
              ENDIF.      "lv_imple_class = '/CPD/CL_PFP_RESTYPE_IACT'.
*               ENDIF.  "<fs_restype> IS NOT INITIAL AND <fs_res> IS INITIAL.
             ENDIF.   "<fs_res> IS ASSIGNED.
            ENDIF.  "IF <fs_restype> IS ASSIGNED.

**** Add code for defalut cost element.
            IF lv_resource_error <> 'X'.
             IF   ls_wbs_cost_detail-lv_cost_element IS  INITIAL AND  ls_wbs_cost_detail-lv_act_type
IS  INITIAL.
*** No cost Element or activity type is maintained at resource type or resource level. read defalut CE
custozing
              IF ls_default_cost_element IS INITIAL OR  ls_default_cost_element-key_figure  <>
lv_key_figure.
               SELECT SINGLE * FROM  /cpd/fc_kf_ceat INTO ls_default_cost_element WHERE
key_figure = lv_key_figure.
              ENDIF.
              IF ls_default_cost_element IS NOT INITIAL AND  ls_default_cost_element-cost_element
IS NOT INITIAL.
               ls_wbs_cost_detail-lv_cost_element  = ls_default_cost_element-cost_element.
              ELSE.
               ls_wbs_cost_detail-lv_act_type = ls_default_cost_element-activity_type.
```

```
          ENDIF.
        ENDIF.

*** 5. Read Month data.
        IF lv_plan_breakdown = '04'.
          ASSIGN COMPONENT 'FISCAL_PERIOD' OF STRUCTURE <fs_data> TO <fs_calmonth>.
          IF <fs_calmonth> IS ASSIGNED.
            ls_wbs_cost_detail-lv_year  = <fs_calmonth>+0(4).
            ls_wbs_cost_detail-lv_period = <fs_calmonth>+4.
          ENDIF.
        ELSE.
          ASSIGN COMPONENT 'CALMONTH' OF STRUCTURE <fs_data> TO <fs_calmonth>.
          IF <fs_calmonth> IS ASSIGNED.
            ls_wbs_cost_detail-lv_year = <fs_calmonth>+0(4).
            ls_wbs_cost_detail-lv_period = <fs_calmonth>+4.
          ENDIF.
        ENDIF.
****6. Read Cost depending upon customizng.
        IF lv_key_figure_cost IS NOT INITIAL.
          ASSIGN COMPONENT 'KEY_FIG_COST' OF STRUCTURE <fs_data> TO <fs_value>.
          IF <fs_value> IS   ASSIGNED.
            ls_wbs_cost_detail-lv_amount = <fs_value>.  " cost
          ENDIF.
        ENDIF.

*** 7. Read Quantity.
        ASSIGN COMPONENT 'QTY' OF STRUCTURE <fs_data> TO <fs_qty>.
        IF <fs_qty> IS   ASSIGNED.
          ls_wbs_cost_detail-lv_qty = <fs_qty>.  " quantity
        ENDIF.
*** 8. Read plan currency
        ASSIGN COMPONENT 'TRANS_CURR' OF STRUCTURE <fs_data> TO <fs_curr>.
        IF <fs_curr> IS   ASSIGNED.
          ls_wbs_cost_detail-lv_curr  = <fs_curr>.  " currency.
        ENDIF.
*** 9. Read UOM
        ASSIGN COMPONENT 'UOM' OF STRUCTURE <fs_data> TO <fs_uom>.
        IF <fs_uom> IS   ASSIGNED.
          ls_wbs_cost_detail-lv_uom  = <fs_uom>.  " currency.
        ENDIF.

*** 10. Read cost center.
        ASSIGN COMPONENT 'COST_CENTER' OF STRUCTURE <fs_data> TO <fs_cost_center>.
        IF <fs_cost_center> IS  ASSIGNED.
          ls_wbs_cost_detail-lv_cost_center = <fs_cost_center>.  " Cost center.
        ENDIF.

**  To check currency of controlling area is empty or not. For cost, convert the currency to
Controlling area currency

        IF <fs_calmonth> IS INITIAL OR  ls_wbs_cost_detail-lv_curr IS INITIAL OR
ls_co_area_detail-currency IS INITIAL OR lv_exrt IS INITIAL.
          ls_message-msgty     = 'E'.
          ls_message-msgid     = '/CPD/PFP_MESSAGES'.
          ls_message-msgno     = '207'.
          ls_message-msgv1     = lv_trans_met_id.
          ls_message-msgv2     = lv_curr_method.
```

```
                ls_message-msgv3     = ls_wbs_cost_detail-lv_wbs.
                IF <fs_calmonth> IS INITIAL.
                 ls_message-msgv4    = <fs_calmonth>.
                ELSEIF ls_wbs_cost_detail-lv_curr IS INITIAL.
                 ls_message-msgv4    = ls_wbs_cost_detail-lv_curr.
                ENDIF.
                APPEND ls_message TO et_message.
               ELSE.
                lv_calmonth = <fs_calmonth>.
                lv_plan_curr = ls_wbs_cost_detail-lv_curr.                " SAP Note-v-2372105
*               IF ls_co_area_detail-currency NE ls_wbs_cost_detail-lv_curr.
*                /cpd/cl_pfp_erp=>curr_conversion(
*                  EXPORTING
*                   iv_monthly  = lv_calmonth
*                   iv_amount   = ls_wbs_cost_detail-lv_amount
*                   iv_fromcurr = ls_wbs_cost_detail-lv_curr
*                   iv_tocurr   = ls_co_area_detail-currency
*                   iv_exrt     = lv_exrt
*                  IMPORTING
*                   ev_amount   = lv_c_amt
*                   et_message  = lt_message
*                    ).
*                IF lt_message IS INITIAL.
**                  lv_ce_amt = lv_c_amt.
*                 ls_wbs_cost_detail-lv_amount  = lv_c_amt. "lv_ce_amt.
*                 ls_wbs_cost_detail-lv_curr    = ls_co_area_detail-currency.
*                ELSE.
*                 LOOP AT lt_message INTO ls_message.
*                  ls_message-msgv1     = lv_curr_method.
*                  ls_message-msgv2     = ls_wbs_cost_detail-lv_wbs.
*                  APPEND ls_message TO et_message.
*                 ENDLOOP.
*                 CLEAR ls_wbs_cost_detail.
*                 CLEAR lt_message.
*                ENDIF.
*               ENDIF.
               ENDIF.
*         endif.

*** 9. Convert/Ignore the UOM.
               IF  ls_wbs_cost_detail IS  NOT INITIAL.
                IF ls_wbs_cost_detail-lv_cost_element IS NOT INITIAL.
** Check if another line item exits for the same WBS plannned on CE which differs only in UOM. If
yes, club by removing  UOM.
                 ls_wbs_cost_detail-lv_uom = ''.
                ELSEIF ls_wbs_cost_detail-lv_act_type IS NOT INITIAL.
** Get the UOM of the activity type & convert the line item UOM to the same.
                 CLEAR : lt_bapiret, ls_activity_type_dt.
                 READ TABLE lt_activity_type_dt INTO ls_activity_type_dt WITH KEY lv_co_area =
lv_co_area lv_activity = ls_wbs_cost_detail-lv_act_type BINARY SEARCH.
                 IF sy-subrc <> 0.
                  CALL FUNCTION 'BAPI_ACTIVITYTYPE_GETDETAIL'
                   EXPORTING
                    controllingarea = lv_co_area
                    activitytype    = ls_wbs_cost_detail-lv_act_type
                   IMPORTING
                    acttypedetail   = ls_act_detail
```

```
          TABLES
            return        = lt_bapiret.
          ls_activity_type_dt-lv_co_area = lv_co_area.
          ls_activity_type_dt-lv_activity = ls_wbs_cost_detail-lv_act_type.
          ls_activity_type_dt-lv_act_unit = ls_act_detail-act_unit.
          INSERT ls_activity_type_dt INTO TABLE lt_activity_type_dt.
        ENDIF.
        IF lt_bapiret IS INITIAL AND ls_activity_type_dt IS NOT INITIAL.
          CALL FUNCTION 'UNIT_CONVERSION_SIMPLE'  "Convert UOM.
            EXPORTING
              input             = ls_wbs_cost_detail-lv_qty
              unit_in           = ls_wbs_cost_detail-lv_uom
              unit_out          = ls_activity_type_dt-lv_act_unit
            IMPORTING
              output            = ls_wbs_cost_detail-lv_qty
            EXCEPTIONS
              conversion_not_found = 1
              division_by_zero    = 2
              input_invalid       = 3
              output_invalid      = 4
              overflow            = 5
              type_invalid        = 6
              units_missing       = 7
              unit_in_not_found   = 8
              unit_out_not_found  = 9
              OTHERS              = 10.
          IF sy-subrc <> 0.
            CLEAR ls_message.
            ls_message-msgty     = 'E'.
            ls_message-msgid     = '/CPD/PFP_MESSAGES'.
            ls_message-msgno     = '218'.
            ls_message-msgv1     = lv_trans_met_id.
            APPEND ls_message TO et_message.
            CLEAR ls_wbs_cost_detail.
          ELSE.
            ls_wbs_cost_detail-lv_uom = ls_activity_type_dt-lv_act_unit.
          ENDIF.
        ELSE. " add error message.
          LOOP AT lt_bapiret INTO ls_bapiret.
            ls_message-msgty     = ls_bapiret-type.
            ls_message-msgid     = ls_bapiret-id.
            ls_message-msgno     = ls_bapiret-number.
            ls_message-msgv1     = ls_bapiret-message_v1.
            ls_message-msgv2     = ls_bapiret-message_v2.
            ls_message-msgv3     = ls_bapiret-message_v3.
            ls_message-msgv4     = ls_bapiret-message_v4.
            APPEND ls_message TO et_message.
            CLEAR: ls_bapiret, ls_message.
          ENDLOOP.
          CLEAR ls_wbs_cost_detail.
        ENDIF. " lt_bapiret IS INITIAL AND ls_act_detail IS NOT INITIAL.
      ENDIF.
    ENDIF.

*** 10. Collect and Insert WBS & Cost ELEMENT/ACT TYPE into wbs_cost_elem to keep track.
    IF ls_wbs_cost_detail IS NOT INITIAL.
      COLLECT: ls_wbs_cost_detail INTO lt_wbs_cost_detail.
```

```
              IF  ls_wbs_cost_detail-lv_cost_element IS NOT INITIAL.
                READ TABLE lt_wbs_cost_elem TRANSPORTING NO FIELDS WITH KEY lv_wbs =
ls_wbs_cost_detail-lv_wbs
                                    lv_cost_element = ls_wbs_cost_detail-lv_cost_element
                                    lv_act_type = ls_wbs_cost_detail-lv_act_type
                                    BINARY SEARCH.
                IF sy-subrc <> 0.
                  CLEAR ls_wbs_cost_elem.
                  ls_wbs_cost_elem-lv_wbs         =  ls_wbs_cost_detail-lv_wbs.
                  ls_wbs_cost_elem-lv_cost_element  = ls_wbs_cost_detail-lv_cost_element.
                  ls_wbs_cost_elem-lv_co_area      = lv_co_area.
                  INSERT ls_wbs_cost_elem INTO TABLE lt_wbs_cost_elem.
                ENDIF.
              ELSE.
                READ TABLE lt_wbs_cost_elem TRANSPORTING NO FIELDS WITH KEY lv_wbs =
ls_wbs_cost_detail-lv_wbs
                lv_cost_element = ls_wbs_cost_detail-lv_cost_element
                lv_act_type = ls_wbs_cost_detail-lv_act_type
                lv_cost_center = ls_wbs_cost_detail-lv_cost_center
                  BINARY SEARCH.
                IF sy-subrc <> 0.
                  CLEAR ls_wbs_cost_elem.
                  ls_wbs_cost_elem-lv_wbs         =  ls_wbs_cost_detail-lv_wbs.
                  ls_wbs_cost_elem-lv_act_type     = ls_wbs_cost_detail-lv_act_type.
                  ls_wbs_cost_elem-lv_co_area      = lv_co_area.
                  ls_wbs_cost_elem-lv_cost_center   = ls_wbs_cost_detail-lv_cost_center.
                  INSERT ls_wbs_cost_elem INTO TABLE lt_wbs_cost_elem.
                ENDIF.
                ls_wbs_cost_detail-lv_act_type = lc_act_type_2000.
                COLLECT: ls_wbs_cost_detail INTO lt_wbs_cost_detail.
                READ TABLE lt_wbs_cost_elem TRANSPORTING NO FIELDS WITH KEY lv_wbs =
ls_wbs_cost_detail-lv_wbs
                lv_cost_element = ls_wbs_cost_detail-lv_cost_element
                lv_act_type = lc_act_type_2000
                lv_cost_center = ls_wbs_cost_detail-lv_cost_center
                  BINARY SEARCH.
                IF sy-subrc <> 0.
                  CLEAR ls_wbs_cost_elem.
                  ls_wbs_cost_elem-lv_wbs         =  ls_wbs_cost_detail-lv_wbs.
                  ls_wbs_cost_elem-lv_act_type     = lc_act_type_2000.
                  ls_wbs_cost_elem-lv_co_area      = lv_co_area.
                  ls_wbs_cost_elem-lv_cost_center   = ls_wbs_cost_detail-lv_cost_center.
                  INSERT ls_wbs_cost_elem INTO TABLE lt_wbs_cost_elem.
                ENDIF.
              ENDIF.  "ls_wbs_cost_detail-LV_COST_ELEMENT IS NOT INITIAL.
            ENDIF.
          ENDIF. "lv_resource_error <> 'X'
        ENDIF. "<fs_wbs_fpid> IS ASSIGNED.
      ENDIF. " controlling area flag
    ENDIF.  "IF <fs_data> IS ASSIGNED.
  ENDLOOP.
ENDIF.  "<fs_plan_data> IS ASSIGNED.
*       ENDIF.

*********************************************************************************
*********************************************************************
```

```
* 11.The table lt_wbs_cost_detail contains the aggregated costs of a WBS for each month,based on
cost element/act type.Loop through this table,pick up the
**    relevant monthly records & call BAPI.
*********************************************************************************
*************************************************************************
       IF lv_co_flag NE 'X' OR iv_ver_del_flag = abap_true.

        IF iv_simulate_transfer EQ abap_false.

***  Deleting all objects irrelevent object from hierarchy
          IF lv_cproj_flag = abap_false.
           DELETE lt_hier_struct_tmp WHERE object_link <> '0WBS'.
          ENDIF.

          IF iv_ver_del_flag = abap_false.
           DELETE lt_hier_struct_tmp WHERE is_authorized <> 'X'.
          ENDIF.
*Delete the planning already exists for the CO Version.
*           CALL METHOD me->delete_planned_data
*            EXPORTING
*              it_hier_struct = lt_hier_struct_tmp
*              it_objid      = lt_objid
*              iv_co_version  = lv_co_version.

          IF iv_ver_del_flag = abap_true.
           CLEAR ls_message.
           ls_message-msgty     = /cpd/cl_pfp_constants=>gc_s.
           ls_message-msgid     = '/CPD/PFP_SC_MESSAGES'.
           ls_message-msgno     = '567'.
           ls_message-msgv1     = lv_trans_met_id.
           APPEND ls_message TO et_message.
          ENDIF.
         ENDIF.
*         IF iv_ver_del_flag = abap_false.
         CLEAR ls_wbs_cost_elem.
         CLEAR ls_wbs_cost_detail.
         SORT lt_wbs_cost_detail BY lv_wbs lv_cost_element lv_act_type lv_year lv_month lv_period.

         LOOP AT lt_wbs_cost_elem INTO ls_wbs_cost_elem.
*** 11.1 Check if either cost element /act type are maintained.
          IF  ls_wbs_cost_elem-lv_cost_element IS INITIAL AND ls_wbs_cost_elem-lv_act_type IS
INITIAL.
           CONCATENATE TEXT-008 ls_wbs_cost_elem-lv_wbs INTO lv_wbs_s SEPARATED BY space.
*** Add error message.
           ls_message-msgty     = 'E'.
           ls_message-msgid     = '/CPD/PFP_MESSAGES'.
           ls_message-msgno     = '206'.
           ls_message-msgv1     = lv_trans_met_id.
           ls_message-msgv2     = lv_wbs_s.
           ls_message-msgv3     = lv_curr_method.
           APPEND ls_message TO et_message.


          ELSEIF ls_wbs_cost_elem-lv_cost_element IS NOT INITIAL.
           CLEAR: lv_period_from,lv_period_to.
*********************************************************************************
**************************
```

```
          " Call BAPI for Cost element.
**********************************************************************************
**************************
****12. Get the starting planned year of the WBS.
          READ TABLE lt_wbs_cost_detail INTO ls_first_wbs_rec WITH KEY lv_wbs =
ls_wbs_cost_elem-lv_wbs  lv_cost_element = ls_wbs_cost_elem-lv_cost_element.
          IF sy-subrc = 0.
           lv_fiscal_year  = ls_first_wbs_rec-lv_year  .
           lv_posting_year = ls_first_wbs_rec-lv_year  .
           lv_count  =  1.
          ENDIF.

          LOOP AT lt_wbs_cost_detail INTO ls_wbs_cost_detail WHERE lv_wbs = ls_wbs_cost_elem-
lv_wbs AND lv_cost_element = ls_wbs_cost_elem-lv_cost_element.
**********************************************************************
           READ TABLE lt_det_val_date INTO ls_det_val_date WITH KEY plan_break_down =
lv_plan_breakdown
                                          lv_year = ls_wbs_cost_detail-lv_year
                                          lv_period = ls_wbs_cost_detail-lv_period
                                          lv_co_area = lv_co_area.
           IF sy-subrc = 0.
            CLEAR lt_valuation_date.
            APPEND LINES OF ls_det_val_date-lt_valuation_date TO lt_valuation_date.
           ELSE.
            ASSIGN ls_wbs_cost_detail TO <fs_line_item>.
            IF <fs_line_item> IS ASSIGNED.
             GET REFERENCE OF <fs_line_item> INTO lr_plan_data.
** Get the posting periods from the fiscal year variant of the controlling area. Pass the date as 1st of
the respective month.
              CLEAR lt_valuation_date.
              CALL METHOD me->determine_valuation_date
               EXPORTING
                is_plan_header     = is_plan_header
                ir_plan_data       = lr_plan_data
                iv_co_area         = ls_wbs_cost_elem-lv_co_area
               IMPORTING
                et_message         = ltab_message
                et_valuation_dates = lt_valuation_date.

              READ TABLE ltab_message TRANSPORTING NO FIELDS WITH KEY msgty = 'E'.
              IF sy-subrc EQ 0.
               APPEND LINES OF ltab_message TO et_message.
               CONTINUE.
              ENDIF.
              ls_det_val_date-plan_break_down = lv_plan_breakdown.
              ls_det_val_date-lv_year = ls_wbs_cost_detail-lv_year.
              ls_det_val_date-lv_period = ls_wbs_cost_detail-lv_period.
              ls_det_val_date-lv_co_area = lv_co_area.
              ls_det_val_date-lt_valuation_date = lt_valuation_date.
              APPEND ls_det_val_date TO lt_det_val_date.
              CLEAR ls_det_val_date.
              UNASSIGN <fs_line_item>.
             ENDIF.
           ENDIF.

           LOOP AT lt_valuation_date INTO ls_valuation_date.
            lv_valuation_date = ls_valuation_date-valuation_date.
```

```
            IF ls_valuation_date-total_days_in_period IS NOT INITIAL.
             lv_amount = ls_wbs_cost_detail-lv_amount * ( ls_valuation_date-days_in_period /
ls_valuation_date-total_days_in_period ).
            ELSE.
             lv_amount =  ls_wbs_cost_detail-lv_amount.
            ENDIF.

**********************************************************************
            IF lv_plan_breakdown = '04'.                    "Note: 2548283
*** Get the posting periods from the fiscal year variant of the controlling area. Pass the date as 1st
of the respective month.
             CALL FUNCTION 'DETERMINE_PERIOD'
              EXPORTING
               date             = lv_valuation_date
               version          = lv_fy_variant
              IMPORTING
               period           = lv_posting_period
               year             = lv_fiscal_year
              EXCEPTIONS
               period_in_not_valid = 1
               period_not_assigned = 2
               version_undefined   = 3
               OTHERS           = 4.
             IF sy-subrc <> 0. " Implement suitable error handling here
              CLEAR ls_message.
              ls_message-msgty     = 'E'.
              ls_message-msgid     = '/CPD/PFP_MESSAGES'.
              ls_message-msgno     = '221'.
              ls_message-msgv1     = lv_trans_met_id.
              ls_message-msgv2     = lv_curr_method.
              ls_message-msgv3     = ls_wbs_cost_elem-lv_wbs.
              CASE  sy-subrc.
               WHEN 1. ls_message-msgv4     = TEXT-003.
               WHEN 2. ls_message-msgv4     = TEXT-004.
               WHEN 3. ls_message-msgv4     = TEXT-005.
              ENDCASE.
              APPEND ls_message TO et_message.
             ENDIF.  "ELSE.
            ELSE.                                   "Note: 2548283
             lv_posting_period = lv_valuation_date+4(2).      "Note: 2548283
             lv_fiscal_year    = lv_valuation_date+0(4).      "Note: 2548283
            ENDIF.
            IF ( lv_fiscal_year =  lv_posting_year ) OR
              ( lv_period_from IS INITIAL AND lv_period_to IS INITIAL ).
             lv_posting_pd = lv_posting_period+1(2).
             CONCATENATE lv_struct_field lv_posting_pd INTO lv_period.
             UNASSIGN <fs_period>.
             ASSIGN COMPONENT lv_period OF STRUCTURE ls_pervalue TO <fs_period>.
             IF <fs_period> IS  ASSIGNED.
              <fs_period> = <fs_period> + lv_amount.
              IF  ( lv_count  = 1 ).      " to get the starting period
               lv_period_from  = lv_posting_period.
              ENDIF.
              lv_count = lv_count + 1.
              lv_period_to  = lv_posting_period.
             ENDIF.
             lv_posting_year  = lv_fiscal_year .
```

```
            ELSE.   " For each WBS, planning data is maintained for more than 1 yr, CALL BAPI for the
current fiscal year .

*** Fill the header info table
              ls_header_info-co_area =  ls_wbs_cost_elem-lv_co_area .
              ls_header_info-fisc_year = lv_posting_year.
              IF lv_period_from IS INITIAL AND lv_period_to IS INITIAL.
                lv_posting_pd = lv_posting_period+1(2).
                CONCATENATE lv_struct_field lv_posting_pd INTO lv_period.
                UNASSIGN <fs_period>.
                ASSIGN COMPONENT lv_period OF STRUCTURE ls_pervalue TO <fs_period>.
                IF <fs_period> IS  ASSIGNED.
                  <fs_period> = <fs_period> + lv_amount.
                  IF  ( lv_count  = 1 ).      " to get the starting period
                    lv_period_from  = lv_posting_period.
                  ENDIF.
                  lv_count = lv_count + 1.
                  lv_period_to  = lv_posting_period.
                ENDIF.
              ENDIF.
              ls_header_info-period_from = lv_period_from.
              ls_header_info-period_to = lv_period_to.
              ls_header_info-plan_currtype = 'T'.
              ls_header_info-version = lv_co_version.

*** Fill CO Object table.
              ls_coobj-wbs_element = ls_wbs_cost_elem-lv_wbs .
              ls_coobj-object_index = 1.
              APPEND ls_coobj TO lt_coobj.

*** FILL INDEX STRUCTURE table
              ls_indextab-value_index = 1.
              ls_indextab-object_index = 1.
              APPEND ls_indextab TO lt_indextab.

*** Fill PERVALUE STRCUTURE table.
              ls_pervalue-value_index = 1.
              ls_pervalue-cost_elem = ls_wbs_cost_elem-lv_cost_element .
              ls_pervalue-trans_curr = lv_plan_curr.
              APPEND ls_pervalue TO lt_pervalue.

*** CALL BAPI
              CALL FUNCTION 'BAPI_COSTACTPLN_POSTPRIMCOST'
                EXPORTING
                  headerinfo    = ls_header_info
                TABLES
                  indexstructure = lt_indextab
                  coobject      = lt_coobj
                  pervalue      = lt_pervalue
                  return        = lt_bapiret.

              IF lt_bapiret IS NOT INITIAL.
*** Append Error Messages.
                LOOP AT lt_bapiret INTO ls_bapiret.
                  ls_message-msgty     = ls_bapiret-type.
                  ls_message-msgid     = ls_bapiret-id.
                  ls_message-msgno     = ls_bapiret-number.
```

```
                ls_message-msgv1     = ls_bapiret-message_v1.
                ls_message-msgv2     = ls_bapiret-message_v2.
                ls_message-msgv3     = ls_bapiret-message_v3.
                ls_message-msgv4     = ls_bapiret-message_v4.
                APPEND ls_message TO et_message.
                CLEAR: ls_bapiret, ls_message.
               ENDLOOP.
              ELSE.
               IF iv_simulate_transfer = ' '.
                CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
                 EXPORTING
                  wait = 'X'.

                CLEAR lv_wbs_s.
                CONCATENATE lv_trans_met_id ':' TEXT-008 ls_wbs_cost_elem-lv_wbs INTO lv_wbs_s
SEPARATED BY space.

                ls_message-msgty     = 'S'.
                ls_message-msgid     = '/CPD/PFP_MESSAGES'.
                ls_message-msgno     = '247'.
                ls_message-msgv1     = lv_wbs_s.
                ls_message-msgv2     = lv_curr_method.
                ls_message-msgv3     = ls_wbs_cost_elem-lv_cost_element.
                ls_message-msgv4     = ls_header_info-fisc_year.

               ELSE.
                CALL FUNCTION 'BAPI_TRANSACTION_ROLLBACK'.
                CONCATENATE lv_trans_met_id ':' lv_curr_method INTO lv_curr_method_s
SEPARATED BY space.
                ls_message-msgty     = 'S'.
                ls_message-msgid     = '/CPD/PFP_MESSAGES'.
                ls_message-msgno     = '248'.
                ls_message-msgv1     = lv_curr_method_s.
                ls_message-msgv2     = ls_wbs_cost_elem-lv_wbs.
                ls_message-msgv3     = ls_wbs_cost_elem-lv_cost_element.
                ls_message-msgv4     = ls_header_info-fisc_year.
               ENDIF.
               APPEND ls_message TO et_message.
               CLEAR: ls_bapiret, ls_message.
              ENDIF.
**** Iniatialize the values
              lv_count = 1.
*               lv_fiscal_year = ls_wbs_cost_detail-lv_year. "Note 2548283
              CLEAR lt_bapiret.
              CLEAR : lt_pervalue , ls_pervalue , lt_indextab , ls_indextab, lt_coobj ,ls_coobj,
ls_header_info.

*** Construct new year's data.
              lv_posting_pd = lv_posting_period+1(2).
              CONCATENATE lv_struct_field lv_posting_pd INTO lv_period.
              UNASSIGN <fs_period>.
              ASSIGN COMPONENT lv_period OF STRUCTURE ls_pervalue TO <fs_period>.
              IF <fs_period> IS  ASSIGNED.
               <fs_period> = <fs_period> + lv_amount.
               IF  ( lv_count  = 1 ).     " to get the starting period
                lv_period_from  = lv_posting_period.
               ENDIF.
```

```
            lv_count = lv_count + 1.
            lv_period_to  =  lv_posting_period.
           ENDIF.
           lv_posting_year  = lv_fiscal_year .
          ENDIF.
         ENDLOOP.
        ENDLOOP.

*** call bapi for the planning data of the current year.

*** Fill the header info table
        ls_header_info-co_area =   ls_wbs_cost_elem-lv_co_area.
        ls_header_info-fisc_year = lv_posting_year.

        IF lv_period_from IS INITIAL AND lv_period_to IS INITIAL.
         lv_posting_pd = lv_posting_period+1(2).
         CONCATENATE lv_struct_field lv_posting_pd INTO lv_period.
         UNASSIGN <fs_period>.
         ASSIGN COMPONENT lv_period OF STRUCTURE ls_pervalue TO <fs_period>.
         IF <fs_period> IS  ASSIGNED.
          <fs_period> = <fs_period> + lv_amount.
          IF  ( lv_count  = 1 ).      " to get the starting period
           lv_period_from  = lv_posting_period.
          ENDIF.
          lv_count = lv_count + 1.
          lv_period_to  = lv_posting_period.
         ENDIF.
        ENDIF.

        ls_header_info-period_from = lv_period_from.
        ls_header_info-period_to = lv_period_to.
        ls_header_info-plan_currtype = 'T'.
        ls_header_info-version = lv_co_version.

*** Fill CO Object table.
        ls_coobj-wbs_element = ls_wbs_cost_elem-lv_wbs .
        ls_coobj-object_index = 1.
        APPEND ls_coobj TO lt_coobj.

*** FILL INDEX STRUCTURE table
        ls_indextab-value_index = 1.
        ls_indextab-object_index = 1.
        APPEND ls_indextab TO lt_indextab.

*** Fill PERVALUE STRCUTURE table.
        ls_pervalue-value_index = 1.
        ls_pervalue-cost_elem = ls_wbs_cost_elem-lv_cost_element .
        ls_pervalue-trans_curr = lv_plan_curr.
        APPEND ls_pervalue TO lt_pervalue.

*** CALL BAPI
        CALL FUNCTION 'BAPI_COSTACTPLN_POSTPRIMCOST'
         EXPORTING
          headerinfo     = ls_header_info
         TABLES
          indexstructure = lt_indextab
          coobject       = lt_coobj
```

```
              pervalue      = lt_pervalue
              return        = lt_bapiret.

           IF lt_bapiret IS NOT INITIAL.
*** Append Error Messages.
            LOOP AT lt_bapiret INTO ls_bapiret.
              ls_message-msgty     = ls_bapiret-type.
              ls_message-msgid     = ls_bapiret-id.
              ls_message-msgno     = ls_bapiret-number.
              ls_message-msgv1     = ls_bapiret-message_v1.
              ls_message-msgv2     = ls_bapiret-message_v2.
              ls_message-msgv3     = ls_bapiret-message_v3.
              ls_message-msgv4     = ls_bapiret-message_v4.
              APPEND ls_message TO et_message.
              CLEAR: ls_bapiret, ls_message.
            ENDLOOP.
           ELSE.
            IF iv_simulate_transfer = ' '.
              CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
                EXPORTING
                  wait = 'X'.
              CLEAR lv_wbs_s.
              CONCATENATE lv_trans_met_id ':' TEXT-008 ls_wbs_cost_elem-lv_wbs INTO lv_wbs_s
SEPARATED BY space.
              ls_message-msgty     = 'S'.
              ls_message-msgid     = '/CPD/PFP_MESSAGES'.
              ls_message-msgno     = '247'.
              ls_message-msgv1     = lv_wbs_s.
              ls_message-msgv2     = lv_curr_method.
              ls_message-msgv3     = ls_wbs_cost_elem-lv_cost_element.
              ls_message-msgv4     = ls_header_info-fisc_year.
            ELSE.
              CALL FUNCTION 'BAPI_TRANSACTION_ROLLBACK'.
              CLEAR lv_curr_method_s.
              CONCATENATE lv_trans_met_id ':' lv_curr_method INTO lv_curr_method_s SEPARATED
BY space.
              ls_message-msgty     = 'S'.
              ls_message-msgid     = '/CPD/PFP_MESSAGES'.
              ls_message-msgno     = '248'.
              ls_message-msgv1     = lv_curr_method_s.
              ls_message-msgv2     = ls_wbs_cost_elem-lv_wbs.
              ls_message-msgv3     = ls_wbs_cost_elem-lv_cost_element.
              ls_message-msgv4     = ls_header_info-fisc_year.
            ENDIF.
            APPEND ls_message TO et_message.
            CLEAR: ls_bapiret, ls_message.

           ENDIF.
**** Iniatialize the values
           CLEAR : lt_pervalue , ls_pervalue , lt_indextab , ls_indextab, lt_coobj ,ls_coobj,
ls_header_info.
           CLEAR lt_bapiret.

          ELSE.    " activity type is maintained.

*********************************************************************************
**************************
```

```
          " Call BAPI for activity type.
*********************************************************************************
**************************

****13. Get the starting planned year of the WBS.
          CLEAR : lv_period_from , lv_period_to ,lv_count.
          READ TABLE lt_wbs_cost_detail INTO ls_first_wbs_rec WITH KEY lv_wbs =
ls_wbs_cost_elem-lv_wbs   lv_act_type  = ls_wbs_cost_elem-lv_act_type
          lv_cost_center = ls_wbs_cost_elem-lv_cost_center.
          IF sy-subrc = 0.
            lv_fiscal_year  = ls_first_wbs_rec-lv_year  .
            lv_posting_year = ls_first_wbs_rec-lv_year  .
            lv_count  =  1.
          ENDIF.
          LOOP AT lt_wbs_cost_detail INTO ls_wbs_cost_detail WHERE lv_wbs = ls_wbs_cost_elem-
lv_wbs AND lv_act_type  = ls_wbs_cost_elem-lv_act_type AND lv_cost_center = ls_wbs_cost_elem-
lv_cost_center.
*********************************************************************
           READ TABLE lt_det_val_date INTO ls_det_val_date WITH KEY plan_break_down =
lv_plan_breakdown
                                    lv_year = ls_wbs_cost_detail-lv_year
                                    lv_period = ls_wbs_cost_detail-lv_period
                                    lv_co_area = lv_co_area.
           IF sy-subrc = 0.
            CLEAR lt_valuation_date.
            APPEND LINES OF ls_det_val_date-lt_valuation_date TO lt_valuation_date.
           ELSE.
            ASSIGN ls_wbs_cost_detail TO <fs_line_item>.
            IF <fs_line_item> IS ASSIGNED.
             GET REFERENCE OF <fs_line_item> INTO lr_plan_data.
** Get the posting periods from the fiscal year variant of the controlling area. Pass the date as 1st of
the respective month.
              REFRESH lt_valuation_date.
              CALL METHOD me->determine_valuation_date
               EXPORTING
                 is_plan_header     = is_plan_header
                 ir_plan_data       = lr_plan_data
                 iv_co_area         = ls_wbs_cost_elem-lv_co_area
               IMPORTING
                 et_message         = ltab_message
                 et_valuation_dates = lt_valuation_date.

              READ TABLE ltab_message TRANSPORTING NO FIELDS WITH KEY msgty = 'E'.
              IF sy-subrc EQ 0.
               APPEND LINES OF ltab_message TO et_message.
               CONTINUE.
              ENDIF.
              ls_det_val_date-plan_break_down = lv_plan_breakdown.
              ls_det_val_date-lv_year = ls_wbs_cost_detail-lv_year.
              ls_det_val_date-lv_period = ls_wbs_cost_detail-lv_period.
              ls_det_val_date-lv_co_area = lv_co_area.
              ls_det_val_date-lt_valuation_date = lt_valuation_date.
              APPEND ls_det_val_date TO lt_det_val_date.
              CLEAR ls_det_val_date.
              UNASSIGN <fs_line_item>.
             ENDIF.
           ENDIF.
```

```
            LOOP AT lt_valuation_date INTO ls_valuation_date.
              lv_valuation_date = ls_valuation_date-valuation_date.
              IF ls_valuation_date-total_days_in_period IS NOT INITIAL.
                lv_qty = ls_wbs_cost_detail-lv_qty * ( ls_valuation_date-days_in_period /
ls_valuation_date-total_days_in_period ).
              ELSE.
                lv_qty =  ls_wbs_cost_detail-lv_qty.
              ENDIF.

**********************************************************************
              IF lv_plan_breakdown = '04'.                    "Note: 2548283
*** Get the posting periods from the fiscal year variant of the controlling area. Pass the date as 1st
of the respective month.
                CALL FUNCTION 'DETERMINE_PERIOD'
                  EXPORTING
                    date             = lv_valuation_date
                    version          = lv_fy_variant
                  IMPORTING
                    period           = lv_posting_period
                    year             = lv_fiscal_year
                  EXCEPTIONS
                    period_in_not_valid = 1
                    period_not_assigned = 2
                    version_undefined   = 3
                    OTHERS           = 4.
                IF sy-subrc <> 0. " Implement suitable error handling here
                  CLEAR ls_message.
                  ls_message-msgty     = 'E'.
                  ls_message-msgid     = '/CPD/PFP_MESSAGES'.
                  ls_message-msgno     = '221'.
                  ls_message-msgv1     = lv_trans_met_id.
                  ls_message-msgv2     = lv_curr_method.
                  ls_message-msgv3     = ls_wbs_cost_elem-lv_wbs.
                  CASE  sy-subrc.
                    WHEN 1. ls_message-msgv4     = TEXT-003.
                    WHEN 2. ls_message-msgv4     = TEXT-004.
                    WHEN 3. ls_message-msgv4     = TEXT-005.
                  ENDCASE.
                  APPEND ls_message TO et_message.
                ENDIF. "  ELSE.
              ELSE.                                 "Note: 2548283
                lv_posting_period = lv_valuation_date+4(2).      "Note: 2548283
                lv_fiscal_year    = lv_valuation_date+0(4).      "Note: 2548283
              ENDIF.
              IF ( lv_fiscal_year =  lv_posting_year ) OR
                 ( lv_period_from IS INITIAL AND lv_period_to IS INITIAL ).
                lv_posting_pd = lv_posting_period+1(2).
                CONCATENATE lv_quant_field lv_posting_pd INTO lv_period.
                UNASSIGN <fs_period>.
                ASSIGN COMPONENT lv_period OF STRUCTURE ls_act_pervalue TO <fs_period>.
                IF <fs_period> IS ASSIGNED.
                  <fs_period> = <fs_period> + lv_qty.
                  IF  ( lv_count  = 1 ).      " to get the starting period
                    lv_period_from  = lv_posting_period.
                  ENDIF.
                  lv_count = lv_count + 1.
```

```
            lv_period_to  = lv_posting_period.
          ENDIF.
          lv_posting_year  = lv_fiscal_year .

        ELSE.   " For each WBS, planning data is maintained for more than 1 yr, CALL BAPI for the current fiscal year .

*** Fill the header info table
          ls_header_info-co_area = ls_wbs_cost_elem-lv_co_area.
          ls_header_info-fisc_year =  lv_posting_year.

          IF lv_period_to IS INITIAL AND lv_period_from IS INITIAL.
           lv_posting_pd = lv_posting_period+1(2).
           CONCATENATE lv_quant_field lv_posting_pd INTO lv_period.
           UNASSIGN <fs_period>.
           ASSIGN COMPONENT lv_period OF STRUCTURE ls_act_pervalue TO <fs_period>.
           IF <fs_period> IS ASSIGNED.
            <fs_period> = <fs_period> + lv_qty.
            IF  ( lv_count  = 1 ).      " to get the starting period
              lv_period_from  = lv_posting_period.
            ENDIF.
            lv_count = lv_count + 1.
            lv_period_to  = lv_posting_period.
           ENDIF.
          ENDIF.

          ls_header_info-period_from = lv_period_from.
          ls_header_info-period_to = lv_period_to.
          ls_header_info-plan_currtype = 'T'.
          ls_header_info-version = lv_co_version.
*** Fill CO Object table.
          ls_act_coobj-wbs_element = ls_wbs_cost_elem-lv_wbs .
          ls_act_coobj-object_index = 1.
          APPEND ls_act_coobj TO lt_act_coobj.

*** FILL INDEX STRUCTURE table
          ls_act_indextab-value_index = 1.
          ls_act_indextab-object_index = 1.
          APPEND ls_act_indextab TO lt_act_indextab.
*** Fill PERVALUE STRCUTURE table.
          ls_act_pervalue-value_index = 1.
          ls_act_pervalue-send_cctr = ls_wbs_cost_elem-lv_cost_center.
          ls_act_pervalue-send_activity = ls_wbs_cost_detail-lv_act_type .
          ls_act_pervalue-unit_of_measure = ls_wbs_cost_detail-lv_uom.
          APPEND ls_act_pervalue TO lt_act_pervalue.
***CALL BAPI.
          CALL FUNCTION 'BAPI_COSTACTPLN_POSTACTINPUT'
           EXPORTING
            headerinfo    = ls_header_info
           TABLES
            indexstructure = lt_act_indextab
            coobject      = lt_act_coobj
            pervalue      = lt_act_pervalue
            return        = lt_bapiret.

          IF lt_bapiret IS NOT INITIAL.
*** Append Error Messages.
```

```
                LOOP AT lt_bapiret INTO ls_bapiret.
                  ls_message-msgty     = ls_bapiret-type.
                  ls_message-msgid     = ls_bapiret-id.
                  ls_message-msgno     = ls_bapiret-number.
                  ls_message-msgv1     = ls_bapiret-message_v1.
                  ls_message-msgv2     = ls_bapiret-message_v2.
                  ls_message-msgv3     = ls_bapiret-message_v3.
                  ls_message-msgv4     = ls_bapiret-message_v4.
                  APPEND ls_message TO et_message.
                  CLEAR: ls_bapiret, ls_message.
                ENDLOOP.
                CLEAR lt_bapiret.
               ELSE.
                IF iv_simulate_transfer = ' '.
                  CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
                    EXPORTING
                      wait = 'X'.
                  CLEAR lv_wbs_s.
                  CONCATENATE lv_trans_met_id ':' TEXT-008 ls_wbs_cost_elem-lv_wbs INTO lv_wbs_s
SEPARATED BY space.

                  ls_message-msgty     = 'S'.
                  ls_message-msgid     = '/CPD/PFP_MESSAGES'.
                  ls_message-msgno     = '247'.
                  ls_message-msgv1     = lv_wbs_s.
                  ls_message-msgv2     = lv_curr_method.
                  ls_message-msgv3     = ls_wbs_cost_elem-lv_act_type.
                  ls_message-msgv4     = ls_header_info-fisc_year.
                ELSE.
                  CALL FUNCTION 'BAPI_TRANSACTION_ROLLBACK'.
                  CLEAR lv_curr_method_s.
                  CONCATENATE lv_trans_met_id ':' lv_curr_method INTO lv_curr_method_s
SEPARATED BY space.
                  ls_message-msgty     = 'S'.
                  ls_message-msgid     = '/CPD/PFP_MESSAGES'.
                  ls_message-msgno     = '248'.
                  ls_message-msgv1     = lv_curr_method_s.
                  ls_message-msgv2     = ls_wbs_cost_elem-lv_wbs.
                  ls_message-msgv3     = ls_wbs_cost_elem-lv_act_type.
                  ls_message-msgv4     = ls_header_info-fisc_year.
                ENDIF.
                APPEND ls_message TO et_message.
                CLEAR: ls_bapiret, ls_message.
               ENDIF.
**** Iniatialize the values
               lv_count = 1.
*                lv_fiscal_year = ls_wbs_cost_detail-lv_year. "Note 2548283
               CLEAR : lt_act_pervalue , ls_act_pervalue , lt_act_indextab , ls_act_indextab,
lt_act_coobj ,ls_act_coobj, ls_header_info.
               CLEAR lt_bapiret.
*** Construct new year's data.
               lv_posting_pd = lv_posting_period+1(2).
               CONCATENATE lv_quant_field lv_posting_pd INTO lv_period.
               UNASSIGN <fs_period>.
               ASSIGN COMPONENT lv_period OF STRUCTURE ls_act_pervalue TO <fs_period>.
               IF <fs_period> IS ASSIGNED.
                <fs_period> = <fs_period> + lv_qty.
```

```
              IF  ( lv_count  = 1 ).      " to get the starting period
                lv_period_from  = lv_posting_period.
              ENDIF.
              lv_count = lv_count + 1.
              lv_period_to  = lv_posting_period.
            ENDIF.
            lv_posting_year  = lv_fiscal_year .
          ENDIF.
        ENDLOOP.
      ENDLOOP.

*** call bapi for the planning data of the current year.
*** Fill the header info table
      ls_header_info-co_area      =  ls_wbs_cost_elem-lv_co_area.
      ls_header_info-fisc_year    =  lv_posting_year.

      IF lv_period_to IS INITIAL AND lv_period_from IS INITIAL.
        lv_posting_pd = lv_posting_period+1(2).
        CONCATENATE lv_quant_field lv_posting_pd INTO lv_period.
        UNASSIGN <fs_period>.
        ASSIGN COMPONENT lv_period OF STRUCTURE ls_act_pervalue TO <fs_period>.
        IF <fs_period> IS ASSIGNED.
          <fs_period> = <fs_period> + lv_qty.
          IF  ( lv_count  = 1 ).
            lv_period_from  = lv_posting_period.
          ENDIF.
          lv_count = lv_count + 1.
          lv_period_to  = lv_posting_period.
        ENDIF.
      ENDIF.
      ls_header_info-period_from   = lv_period_from.
      ls_header_info-period_to     = lv_period_to.
      ls_header_info-plan_currtype = 'T'.
      ls_header_info-version       = lv_co_version.

*** Fill CO Object table.
      ls_act_coobj-wbs_element = ls_wbs_cost_elem-lv_wbs .
      ls_act_coobj-object_index = 1.
      APPEND ls_act_coobj TO lt_act_coobj.

*** FILL INDEX STRUCTURE table
      ls_act_indextab-value_index = 1.
      ls_act_indextab-object_index = 1.
      APPEND ls_act_indextab TO lt_act_indextab.

*** Fill PERVALUE STRCUTURE table.
      ls_act_pervalue-value_index = 1.
      ls_act_pervalue-send_cctr = ls_wbs_cost_elem-lv_cost_center.
      ls_act_pervalue-send_activity = ls_wbs_cost_detail-lv_act_type .
      ls_act_pervalue-unit_of_measure = ls_wbs_cost_detail-lv_uom.
      APPEND ls_act_pervalue TO lt_act_pervalue.

***CALL BAPI.
      CALL FUNCTION 'BAPI_COSTACTPLN_POSTACTINPUT'
        EXPORTING
          headerinfo     = ls_header_info
        TABLES
```

```
              indexstructure = lt_act_indextab
              coobject       = lt_act_coobj
              pervalue       = lt_act_pervalue
              return         = lt_bapiret.

            IF lt_bapiret IS NOT INITIAL.
*** Append Error Messages.
              LOOP AT lt_bapiret INTO ls_bapiret.
                ls_message-msgty     = ls_bapiret-type.
                ls_message-msgid     = ls_bapiret-id.
                ls_message-msgno     = ls_bapiret-number.
                ls_message-msgv1     = ls_bapiret-message_v1.
                ls_message-msgv2     = ls_bapiret-message_v2.
                ls_message-msgv3     = ls_bapiret-message_v3.
                ls_message-msgv4     = ls_bapiret-message_v4.
                APPEND ls_message TO et_message.
                CLEAR: ls_bapiret, ls_message.
              ENDLOOP.
            ELSE.
              IF iv_simulate_transfer = ' '.
                CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
                  EXPORTING
                    wait = 'X'.
                CLEAR lv_wbs_s.
                CONCATENATE lv_trans_met_id ':' TEXT-008 ls_wbs_cost_elem-lv_wbs INTO lv_wbs_s
SEPARATED BY space.
                ls_message-msgty     = 'S'.
                ls_message-msgid     = '/CPD/PFP_MESSAGES'.
                ls_message-msgno     = '247'.
                ls_message-msgv1     = lv_wbs_s.
                ls_message-msgv2     = lv_curr_method.
                ls_message-msgv3     = ls_wbs_cost_elem-lv_act_type.
                ls_message-msgv4     = ls_header_info-fisc_year.
              ELSE.
                CALL FUNCTION 'BAPI_TRANSACTION_ROLLBACK'.
                CLEAR lv_curr_method_s.
                CONCATENATE lv_trans_met_id ':' lv_curr_method INTO lv_curr_method_s SEPARATED
BY space.
                ls_message-msgty     = 'S'.
                ls_message-msgid     = '/CPD/PFP_MESSAGES'.
                ls_message-msgno     = '248'.
                ls_message-msgv1     = lv_curr_method_s.
                ls_message-msgv2     = ls_wbs_cost_elem-lv_wbs.
                ls_message-msgv3     = ls_wbs_cost_elem-lv_act_type.
                ls_message-msgv4     = ls_header_info-fisc_year.
              ENDIF.
              APPEND ls_message TO et_message.
              CLEAR: ls_bapiret, ls_message.
            ENDIF.
**** Iniatialize the values
            CLEAR : lt_act_pervalue , ls_act_pervalue , lt_act_indextab , ls_act_indextab,
lt_act_coobj ,ls_act_coobj, ls_header_info.
            CLEAR lt_bapiret.

            IF iv_simulate_transfer NE ' '.
              CALL FUNCTION 'BAPI_TRANSACTION_ROLLBACK'.
            ENDIF.
```

```
          ENDIF.
         ENDLOOP.
*          ENDIF.
        ELSE.
          ls_message-msgty     = 'E'.
          ls_message-msgid     = '/CPD/PFP_MESSAGES'.
          ls_message-msgno     = '199'.
          ls_message-msgv1     = lv_trans_met_id.
          ls_message-msgv2     = lv_curr_method.
          APPEND ls_message TO et_message.
          CLEAR ls_message.
        ENDIF.  " lv_co_ver is NE 'X'.
      ENDIF.  "lv_key_figure_cost IS NOT INITIAL .

    ELSE.   " co_version is intial
      ls_message-msgty     = 'E'.
      ls_message-msgid     = '/CPD/PFP_MESSAGES'.
      ls_message-msgno     = '153'.
      ls_message-msgv1     = lv_trans_met_id.
      ls_message-msgv2     = lv_curr_method.
      APPEND ls_message TO et_message.

    ENDIF.  "   IV_CO_VERSION IS NOT INITIAL .
* endif. " controlling area checking.
   ENDIF.
 ENDMETHOD.


* <SIGNATURE>---------------------------------------------------------------------------------------+
* | Instance Public Method
ZCL_CON_CPM_PFP_WBS_COST_UPD->/CPD/IF_PFP_DOWNPORT_METHOD~GET_DOWNPORT_REL
EVANT_FIELDS
* +-------------------------------------------------------------------------------------------------+
* | [<---] ET_DOWNPORT_FIELDS           TYPE       /CPD/T_PFP_DOWNPORT_FIELDS
* +-------------------------------------------------------------------------------------</SIGNATURE>
 METHOD /CPD/IF_PFP_DOWNPORT_METHOD~GET_DOWNPORT_RELEVANT_FIELDS.

**  The method provides the fileds for F4 help in the custmozing for defining transfer/downport
method.
** It retruns the Fileds for which values are supplied in customizing, for relevant importing
structures for calling 'BAPI_COSTACTPLN_POSTPRIMCOST'.


   DATA : lt_ddic_fields       TYPE TABLE OF dfies,
          lt_ddic_struct_fields TYPE TABLE OF dfies,
          ls_ddic_field        TYPE dfies.

   DATA :  ls_transfer_fields    TYPE /cpd/s_pfp_downport_fields.


   CALL FUNCTION 'DDIF_FIELDINFO_GET'
    EXPORTING
     tabname   = '/CPD/S_PFP_SC_WBS_CO'
    TABLES
     dfies_tab = lt_ddic_fields.

   LOOP AT lt_ddic_fields INTO ls_ddic_field.
```

```
    ls_transfer_fields-data_type = ls_ddic_field-datatype.
    ls_transfer_fields-transfer_method_field = ls_ddic_field-fieldname .
    ls_transfer_fields-fieldtext = ls_ddic_field-fieldtext.
    ls_transfer_fields-length = ls_ddic_field-leng.
    APPEND ls_transfer_fields  TO et_downport_fields.
   ENDLOOP.

 ENDMETHOD.


* <SIGNATURE>---------------------------------------------------------------------------------------+
* | Static Public Method
ZCL_CON_CPM_PFP_WBS_COST_UPD=>/CPD/IF_PFP_DOWNPORT_METHOD~GET_INPUT_STRUCTU
RE
* +---------------------------------------------------------------------------------------------------+
* | [<-()] EV_INPUT                 TYPE       CHAR30
* +-------------------------------------------------------------------------------------</SIGNATURE>
 METHOD /CPD/IF_PFP_DOWNPORT_METHOD~GET_INPUT_STRUCTURE.

   ev_input = '/CPD/S_PFP_SC_WBS_CO'.

 ENDMETHOD.


* <SIGNATURE>---------------------------------------------------------------------------------------+
* | Instance Public Method
ZCL_CON_CPM_PFP_WBS_COST_UPD->/CPD/IF_PFP_DOWNPORT_METHOD~PERFORM_COPY
* +---------------------------------------------------------------------------------------------------+
* | [<-()] RV_TO_COPY               TYPE       BOOLE_D
* +-------------------------------------------------------------------------------------</SIGNATURE>
 METHOD /CPD/IF_PFP_DOWNPORT_METHOD~PERFORM_COPY.
 ENDMETHOD.


* <SIGNATURE>---------------------------------------------------------------------------------------+
* | Instance Private Method ZCL_CON_CPM_PFP_WBS_COST_UPD->DELETE_PLANNED_DATA
* +---------------------------------------------------------------------------------------------------+
* | [--->] IT_HIER_STRUCT           TYPE       /CPD/T_SC_PFP_EXECUTION_HIER
* | [--->] IT_OBJID                 TYPE       TY_T_OBJID
* | [--->] IV_CO_VERSION            TYPE       VERSN
* +-------------------------------------------------------------------------------------</SIGNATURE>
 METHOD DELETE_PLANNED_DATA.
   DATA :
    lt_objid        TYPE ty_t_objid,
    ls_hier_struct   TYPE /cpd/s_pfp_execution_hier,
    lt_cosp_tab     TYPE TABLE OF cospa,
    ls_cosp_tab     TYPE cospa,
    lt_coss_tab     TYPE TABLE OF cossa,
    ls_coss_tab     TYPE cossa,
    ls_objid        TYPE ty_objid,
    lv_proj_guid     TYPE dpr_tv_entity_guid_co,
    lt_hierarchy_tmp TYPE iaom_cprojects_tt_guid,
    lt_hierarchy     TYPE iaom_cprojects_tt_guid,
    ls_hierarchy     TYPE iaom_cprojects_ts_guid,
    lv_type         TYPE beknz,
    lt_del_objnr     TYPE TABLE OF objlist_ps,
    ls_del_objnr     TYPE objlist_ps.
```

```
  CONSTANTS: lco_d TYPE beknz VALUE 'S'.

  lt_objid[] = it_objid[].
  READ TABLE it_hier_struct TRANSPORTING NO FIELDS WITH KEY object_link ='0DPO'.
  IF sy-subrc = 0.
***     Call BAPI to read attached WBS with cproj
    LOOP AT it_hier_struct INTO ls_hier_struct WHERE object_link = '0DPO'.
     CLEAR lv_proj_guid.
     lv_proj_guid = ls_hier_struct-object_id.

     CALL FUNCTION 'IAOM_CPRO_MISC_RFC_GET_EL_LIST'
      EXPORTING
       iv_cpro_project_guid      = lv_proj_guid
      IMPORTING
       et_hierarchy             = lt_hierarchy
      EXCEPTIONS
       cpro_project_unknown      = 1
       no_cpro_project_definition = 2
       OTHERS                   = 3.
     IF sy-subrc <> 0.
* Implement suitable error handling here
     ELSE.
      APPEND LINES OF lt_hierarchy TO lt_hierarchy_tmp.
      REFRESH lt_hierarchy.
     ENDIF.

    ENDLOOP.
***   Adding left WBS objects to the Object table
    LOOP AT lt_hierarchy_tmp INTO ls_hierarchy.
     ls_objid-objnr = ls_hierarchy-objnr.
     INSERT ls_objid INTO TABLE lt_objid.
    ENDLOOP.
  ELSE.
    LOOP AT it_hier_struct INTO ls_hier_struct.
     ls_objid-objnr = ls_hier_struct-object_id.
     INSERT ls_objid INTO TABLE lt_objid.
    ENDLOOP.
  ENDIF.

  IF lt_objid IS NOT INITIAL.
    REFRESH: lt_coss_tab, lt_cosp_tab.
***Reading the planning from Database and deleting them. COSS contains data of secondry cost
elements
    SELECT
    lednr
    objnr
    gjahr
    wrttp
    versn
    kstar
    hrkft
    vrgng
    parob
    uspob
    beknz
    twaer
```

```
    FROM coss             ##DB_FEATURE_MODE[TABLE_LEN_MAX1]
    INTO CORRESPONDING FIELDS OF TABLE lt_coss_tab
    FOR ALL ENTRIES IN lt_objid
    WHERE objnr = lt_objid-objnr AND versn = iv_co_version.
    IF sy-subrc = 0.
     CALL FUNCTION 'K_COSSA_DELETE'
      TABLES
       cossa_tab = lt_coss_tab.

     CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
      EXPORTING
       wait = 'X'.
    ENDIF.

***Reading the planning from Database and deleting them.
    SELECT
    lednr
    objnr
    gjahr
    wrttp
    versn
    kstar
    hrkft
    vrgng
    vbund
    pargb
    beknz
    twaer
    perbl
    FROM cosp             ##DB_FEATURE_MODE[TABLE_LEN_MAX1]
    INTO CORRESPONDING FIELDS OF TABLE lt_cosp_tab
    FOR ALL ENTRIES IN lt_objid
    WHERE objnr = lt_objid-objnr AND versn = iv_co_version.
    IF sy-subrc = 0.
     lv_type = lco_d.
     DELETE lt_cosp_tab WHERE beknz <> lv_type.
     IF lt_cosp_tab IS NOT INITIAL.
      CALL FUNCTION 'K_COSPA_DELETE'
       TABLES
        cospa_tab = lt_cosp_tab.

      CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
       EXPORTING
        wait = 'X'.
     ENDIF.
    ENDIF.
****delete entries from RPSCO table     " Note 2493656
    LOOP AT lt_objid INTO ls_objid.
     ls_del_objnr-objnr = ls_objid-objnr.
     APPEND ls_del_objnr TO lt_del_objnr.
    ENDLOOP.
    IF lt_del_objnr[] IS NOT INITIAL.
     CALL FUNCTION 'HFPV_DELETE_RPSCO_V2'
      TABLES
       del_objnr = lt_del_objnr.
    ENDIF.
   ENDIF.
```

```
 ENDMETHOD.


* <SIGNATURE>---------------------------------------------------------------------------------------+
* | Instance Protected Method ZCL_CON_CPM_PFP_WBS_COST_UPD->
DETERMINE_VALUATION_DATE
* +---------------------------------------------------------------------------------------------------+
* | [--->] IR_PLAN_DATA                TYPE REF TO DATA
* | [--->] IS_PLAN_HEADER              TYPE       /CPD/S_PFP_PLAN_HEADER_D
* | [--->] IV_CO_AREA                  TYPE       KOKRS
* | [<---] ET_MESSAGE                  TYPE       /CPD/T_MESSAGE
* | [<---] ET_VALUATION_DATES          TYPE       /CPD/T_PFP_VALUTION_DATES
* +---------------------------------------------------------------------------------------</SIGNATURE>
  METHOD DETERMINE_VALUATION_DATE.
*** This method determines the valuation date. The valuation date is determined in -
*** 1. IF it is fiscal plannig, then the it is determined under which CO fiscal periods does the CPM
fiscal falls.

****   Add buffer tables for -- Controlling area , activity type details , PS Project, netwrok activity
details
   TYPES : BEGIN OF ts_controlling_area,
           lv_co_area    TYPE  kokrs,
           lv_fy_variant TYPE  periv,
          END OF ts_controlling_area.

   DATA:lr_pfp_erp          TYPE REF TO /cpd/cl_pfp_erp,
        lt_co_period_mapping TYPE /cpd/t_pfp_sc_co_fis_per_map,
        ls_co_period_mapping TYPE /cpd/s_pfp_sc_co_fis_per_map,
        ls_valuation_date    TYPE /cpd/s_pfp_valution_dates,
        lv_valuation_date    TYPE sy-datum,
        lv_cpm_period_sd     TYPE sy-datum,
        lv_cpm_period_ed     TYPE sy-datum,
        lv_co_period_sd      TYPE sy-datum,
        lv_co_period_ed      TYPE sy-datum,
        lv_co_period         TYPE /cpd/pfp_fisc_period,
        lv_co_period1        TYPE /cpd/pfp_fisc_period,
        ls_plan_header       TYPE /cpd/s_pfp_plan_header,
        ls_co_area_detail    TYPE bapi0004_2,
        ls_coabapiret        TYPE bapireturn,
        lv_fy_variant        TYPE periv,
        lv_fy_var            TYPE string,
        lv_count             TYPE i VALUE 1,
        lv_total_days        TYPE i,
        lv_plan_breakdown    TYPE /cpd/pfp_period,
        lv_cpm_period        TYPE /cpd/pfp_fisc_period,
        lv_mp_fisvar         TYPE periv,
        lv_bool              TYPE boole_d.

   DATA: lt_controlling_area_dt TYPE  TABLE OF   ts_controlling_area,
         ls_controlling_area_dt TYPE           ts_controlling_area,
         lv_method              TYPE           string,
         ls_message             TYPE           /cpd/s_message.


   FIELD-SYMBOLS :
    <fs_plan_data>      TYPE any,
    <fv_year>           TYPE any,
```

```
    <fv_period>       TYPE any,
    <fv_wbs>          TYPE  ps_posid,
    <fs_valuation_date> TYPE /cpd/s_pfp_valution_dates.

   CLEAR: et_valuation_dates, ls_controlling_area_dt.
   CONCATENATE TEXT-006 TEXT-001 INTO lv_method.

   ASSIGN ir_plan_data->* TO <fs_plan_data>.
   IF <fs_plan_data> IS NOT ASSIGNED.
    RETURN.
   ENDIF.

   lv_plan_breakdown    = /cpd/cl_pfp_erp=>mv_period.

   ASSIGN COMPONENT 'LV_WBS' OF STRUCTURE <fs_plan_data> TO <fv_wbs>.
   IF <fv_wbs> IS ASSIGNED.
    CASE lv_plan_breakdown.
     WHEN '04'.                        "For Fiscal Year Planning

       CALL METHOD /cpd/cl_pfp_erp=>get_instance
        IMPORTING
          er_pfp_erp = lr_pfp_erp.    " PFP ERP integration

       lt_co_period_mapping = lr_pfp_erp->mt_co_period_mapping.
       lv_mp_fisvar        = /cpd/cl_pfp_erp=>mv_plan_fiscvar.

       ASSIGN COMPONENT 'LV_YEAR'   OF STRUCTURE <fs_plan_data> TO <fv_year>.
       ASSIGN COMPONENT 'LV_PERIOD' OF STRUCTURE <fs_plan_data> TO <fv_period>.

       IF <fv_year> IS ASSIGNED AND <fv_period> IS ASSIGNED.
        CONCATENATE  <fv_year> <fv_period> INTO lv_cpm_period.

        CLEAR lv_total_days.
        LOOP AT lt_co_period_mapping INTO ls_co_period_mapping WHERE co_area = iv_co_area
AND  cpm_period  = lv_cpm_period.
          CLEAR ls_valuation_date.
          ls_valuation_date-valuation_date          = ls_co_period_mapping-start_date.
          ls_valuation_date-days_in_period          = ls_co_period_mapping-end_date -
ls_co_period_mapping-start_date + 1.
          lv_total_days   = lv_total_days + ls_valuation_date-days_in_period .
          APPEND ls_valuation_date TO et_valuation_dates.
        ENDLOOP.

        LOOP AT et_valuation_dates ASSIGNING <fs_valuation_date>.
          IF <fs_valuation_date> IS ASSIGNED.
            <fs_valuation_date>-total_days_in_period = lv_total_days.
          ENDIF.
        ENDLOOP.
       ENDIF.

       "If Record for given selection not exists in Instance table
       IF et_valuation_dates IS INITIAL AND <fv_year> IS ASSIGNED AND <fv_period> IS ASSIGNED.

*** Determine the Fiscal Varaint of the controlling area.
        READ TABLE lt_controlling_area_dt INTO ls_controlling_area_dt WITH KEY  lv_co_area =
iv_co_area.
        IF sy-subrc <> 0.
```

```
          CALL FUNCTION 'BAPI_CONTROLLINGAREA_GETDETAIL'
            EXPORTING
              controllingareaid     = iv_co_area
            IMPORTING
              controllingarea_detail = ls_co_area_detail
              return                 = ls_coabapiret.
          IF ls_coabapiret IS INITIAL AND ls_co_area_detail IS NOT INITIAL.
            lv_fy_variant = ls_co_area_detail-fy_variant.
            ls_controlling_area_dt-lv_fy_variant = ls_co_area_detail-fy_variant.
            ls_controlling_area_dt-lv_co_area = iv_co_area .
            APPEND ls_controlling_area_dt TO lt_controlling_area_dt.
          ENDIF.
        ELSE.
          lv_fy_variant = ls_controlling_area_dt-lv_fy_variant.
        ENDIF. "sy-subrc <> 0



****** Determine the start and end date of the CPM period.
        CALL METHOD /cpd/cl_pfp_po_services=>get_start_end_date_fiscper
          EXPORTING
            iv_fiscper     = lv_cpm_period
            iv_variant     = lv_mp_fisvar
          IMPORTING
            ev_start_date  = lv_cpm_period_sd
            ev_end_date    = lv_cpm_period_ed
          EXCEPTIONS
            input_false    = 1
            t009_notfound  = 2
            t009b_notfound = 3
            OTHERS         = 4.
        IF sy-subrc <> 0.
          CLEAR ls_message.
          ls_message-msgty     = /cpd/cl_pfp_constants=>gc_e.
          ls_message-msgid     = /cpd/cl_pfp_constants=>gc_msg_class_name.
          ls_message-msgno     = '412'.
          ls_message-msgv1     = lv_method . "methos
          ls_message-msgv2     = <fv_wbs>.   "wbs
          ls_message-msgv3     = lv_cpm_period. "period
          ls_message-msgv4     = lv_mp_fisvar.   "fiscal Variant.

          APPEND ls_message TO et_message.
        ELSE.
***** Check if the start and end date of the CPM period falls under same CO Fiscal period .
          CALL METHOD /cpd/cl_pfp_po_services=>determine_date_to_period
            EXPORTING
              iv_date       = lv_cpm_period_sd   "start date
              iv_variant    = lv_fy_variant
            IMPORTING
              ev_period     = lv_co_period
            EXCEPTIONS
              input_false    = 1
              t009_notfound  = 2
              t009b_notfound = 3
              OTHERS         = 4.
          IF sy-subrc <> 0.
            CLEAR ls_message.
```

```
            ls_message-msgty     = /cpd/cl_pfp_constants=>gc_e.
            ls_message-msgid     = /cpd/cl_pfp_constants=>gc_msg_class_name.
            ls_message-msgno     = '413'.
            ls_message-msgv1     = lv_method . "methos
            ls_message-msgv2     = <fv_wbs>.   "wbs
            ls_message-msgv3     =   lv_cpm_period_sd. "period
            ls_message-msgv4     = lv_fy_variant.   "fiscal Variant.

            APPEND ls_message TO et_message.
            EXIT.
          ENDIF.

          CALL METHOD /cpd/cl_pfp_po_services=>determine_date_to_period
            EXPORTING
              iv_date       = lv_cpm_period_ed  "end date
              iv_variant    = lv_fy_variant
            IMPORTING
              ev_period     = lv_co_period1
            EXCEPTIONS
              input_false   = 1
              t009_notfound  = 2
              t009b_notfound = 3
              OTHERS        = 4.
          IF sy-subrc <> 0.
            CLEAR ls_message.
            ls_message-msgty     = /cpd/cl_pfp_constants=>gc_e.
            ls_message-msgid     = /cpd/cl_pfp_constants=>gc_msg_class_name.
            ls_message-msgno     = '413'.
            ls_message-msgv1     = lv_method . "methos
            ls_message-msgv2     = <fv_wbs>.   "wbs
            ls_message-msgv3     =   lv_cpm_period_sd. "period
            ls_message-msgv4     = lv_fy_variant.   "fiscal Variant.

            APPEND ls_message TO et_message.
            EXIT.
          ENDIF.

          IF lv_co_period = lv_co_period1.
**** The CPM period falls under one CO  fiscal period
            CLEAR ls_valuation_date.
            IF is_plan_header-start_date GT lv_cpm_period_sd.
              lv_cpm_period_sd = is_plan_header-start_date.
            ENDIF.

            IF is_plan_header-end_date LT lv_cpm_period_ed.
              lv_cpm_period_ed = is_plan_header-end_date.
            ENDIF.

            ls_valuation_date-valuation_date    =   lv_cpm_period_sd.
            APPEND ls_valuation_date TO et_valuation_dates.

*** Append this record to the mt table.
            CLEAR ls_co_period_mapping.
            ls_co_period_mapping-co_period     = lv_co_period.
            ls_co_period_mapping-cpm_period    = lv_cpm_period.
            ls_co_period_mapping-start_date    = lv_cpm_period_sd.
            ls_co_period_mapping-end_date      = lv_cpm_period_ed.
```

```
          ls_co_period_mapping-no_of_days    = lv_cpm_period_ed - lv_cpm_period_sd + 1.
          ls_co_period_mapping-co_area       = iv_co_area.

          READ TABLE lt_co_period_mapping WITH KEY co_area = iv_co_area     co_period =
lv_co_period    cpm_period  = lv_cpm_period TRANSPORTING NO FIELDS.
          IF sy-subrc <> 0.
            APPEND ls_co_period_mapping TO lt_co_period_mapping.
          ENDIF.
         ELSE.
*** The CPM perid falls under different CO periods.
***   Now start by finding the start and END dates of the CO period under which lv_cpm_period_sd
falls

****** Determine the start and end date of the CO period.

          WHILE lv_co_period <= lv_co_period1.
            CLEAR : lv_co_period_sd , lv_co_period_ed ,ls_co_period_mapping.

            CALL METHOD /cpd/cl_pfp_po_services=>get_start_end_date_fiscper
              EXPORTING
                iv_fiscper     = lv_co_period
                iv_variant     = lv_fy_variant
              IMPORTING
                ev_start_date  = lv_co_period_sd
                ev_end_date    = lv_co_period_ed
              EXCEPTIONS
                input_false    = 1
                t009_notfound  = 2
                t009b_notfound = 3
                OTHERS         = 4.
            IF sy-subrc <> 0.
              CLEAR ls_message.
              ls_message-msgty     = /cpd/cl_pfp_constants=>gc_e.
              ls_message-msgid     = /cpd/cl_pfp_constants=>gc_msg_class_name.
              ls_message-msgno     = '412'.
              ls_message-msgv1     = lv_method . "methos
              ls_message-msgv2     = <fv_wbs>.   "wbs
              ls_message-msgv3     = lv_co_period. "period
              ls_message-msgv4     = lv_fy_variant.   "fiscal Variant.

              APPEND ls_message TO et_message.
              EXIT.
            ENDIF.

            ls_co_period_mapping-co_period     = lv_co_period.
            ls_co_period_mapping-cpm_period    = lv_cpm_period.

            IF lv_count = 1.
              ls_co_period_mapping-start_date = lv_cpm_period_sd.
            ELSE.
              ls_co_period_mapping-start_date = lv_co_period_sd.
            ENDIF.

            lv_count  = lv_count + 1.
            IF lv_co_period_ed > lv_cpm_period_ed.
              ls_co_period_mapping-end_date   = lv_cpm_period_ed.
            ELSE.
```

```
            ls_co_period_mapping-end_date   = lv_co_period_ed.
           ENDIF.

           ls_co_period_mapping-no_of_days =  ls_co_period_mapping-end_date -
ls_co_period_mapping-start_date + 1.
           ls_co_period_mapping-co_area     = iv_co_area.

           READ TABLE lt_co_period_mapping WITH KEY co_area = iv_co_area     co_period =
lv_co_period    cpm_period  = lv_cpm_period TRANSPORTING NO FIELDS.
           IF sy-subrc <> 0.
            APPEND ls_co_period_mapping TO lt_co_period_mapping.
           ENDIF.

*** Get the next period
           lv_fy_var = lv_fy_variant.
           CALL FUNCTION '/CPD/PFP_GET_NEXT_FISC_PERIOD'
            EXPORTING
              iv_fvariant    = lv_fy_var
              iv_fiscper     = lv_co_period
            IMPORTING
              ev_next_period = lv_co_period.

***Fill return table
           CLEAR ls_valuation_date.
           ls_valuation_date-valuation_date          =   ls_co_period_mapping-start_date.
           ls_valuation_date-days_in_period          =   ls_co_period_mapping-no_of_days.
           lv_total_days   = lv_total_days + ls_valuation_date-days_in_period .
           APPEND ls_valuation_date TO et_valuation_dates.
          ENDWHILE.
         ENDIF.  "lv_co_period = lv_co_period1.
        ENDIF.

        LOOP AT et_valuation_dates ASSIGNING <fs_valuation_date>.
         IF <fs_valuation_date> IS ASSIGNED.
          <fs_valuation_date>-total_days_in_period = lv_total_days.
         ENDIF.
        ENDLOOP.

        lr_pfp_erp->mt_co_period_mapping = lt_co_period_mapping.
       ENDIF.

      WHEN OTHERS.
       CLEAR:lv_valuation_date.
       ASSIGN COMPONENT 'LV_YEAR'  OF STRUCTURE <fs_plan_data> TO <fv_year>.
       ASSIGN COMPONENT 'LV_PERIOD' OF STRUCTURE <fs_plan_data> TO <fv_period>.

       IF <fv_year> IS ASSIGNED AND <fv_period> IS ASSIGNED.
        CONCATENATE <fv_year>  <fv_period> '01' INTO lv_valuation_date.
        ls_valuation_date-valuation_date          =   lv_valuation_date.
        APPEND ls_valuation_date TO et_valuation_dates.
       ENDIF.
    ENDCASE.
   ENDIF. "  <fv_wbs> IS ASSIGNED.
 ENDMETHOD.
ENDCLASS.
```

```
SELECT team~engagementprojectuuid as projuuid,
   teamrole~engagementprojectteamrole AS teamrole,
   STRING_AGG( projectmember~employmentinternalid,';' )  AS employmentinternalids ,
   count( projectmember~employmentinternalid ) as counts
 FROM i_engagementprojectteam AS team
  LEFT OUTER JOIN i_engagementprojectteamrole AS teamrole ON
team~engagementprojectteamuuid = teamrole~engagementprojectteamuuid
  LEFT OUTER JOIN i_engmtprojteammember AS teammember  ON
teamrole~engagementprojectteamroleuuid = teammember~engagementprojectteamroleuuid
  LEFT OUTER JOIN i_engagementprojectmember AS projectmember  ON
teammember~engagementprojectmemberuuid = projectmember~engagementprojectmemberuuid
AND projectmember~BusinessPartnerMemberType = 'BUP003'
  where teamrole~engagementprojectteamrole in ( 'Z003','Z004' )
  group by team~engagementprojectuuid,teamrole~engagementprojectteamrole
  .



SELECT
        team~engagementprojectuuid AS projuuid,
        teamrole~engagementprojectteamrole AS teamrole,
        STRING_AGG( projectmember~employmentinternalid,';' )  AS employmentinternalids ,
        COUNT( projectmember~employmentinternalid ) AS counts
      FROM i_engagementprojectteam AS team
      INNER JOIN @lt_proj as proj on team~engagementprojectuuid = proj~db_key
      LEFT OUTER JOIN i_engagementprojectteamrole AS teamrole ON
team~engagementprojectteamuuid = teamrole~engagementprojectteamuuid
      LEFT OUTER JOIN i_engmtprojteammember AS teammember  ON
teamrole~engagementprojectteamroleuuid = teammember~engagementprojectteamroleuuid
      LEFT OUTER JOIN i_engagementprojectmember AS projectmember  ON
teammember~engagementprojectmemberuuid = projectmember~engagementprojectmemberuuid
AND projectmember~BusinessPartnerMemberType = 'BUP003'
      WHERE teamrole~engagementprojectteamrole IN ( @lc_roleid_z003,@lc_roleid_z004 )
      GROUP BY team~engagementprojectuuid,teamrole~engagementprojectteamrole
      INTO TABLE @DATA(lt_duptymc).
     IF sy-subrc = 0.
       sort lt_duptymc by projuuid teamrole.
     ENDIF.



     LOOP AT GROUP ls_proj INTO DATA(ls_proj_sub).
*     Get deputy project controller name
       IF ls_proj_sub-deputypc IS NOT INITIAL.
         READ TABLE lt_mp_bp_dpc INTO DATA(ls_mp_bp_dpc) WITH KEY businesspartner =
ls_proj_sub-deputypc.
         IF sy-subrc EQ 0.
           IF ls_data-deputypc IS NOT INITIAL AND ls_data-deputypc NS ls_mp_bp_dpc-
businesspartnerfullname.
             CONCATENATE ls_data-deputypc ';' ls_mp_bp_dpc-businesspartnerfullname INTO ls_data-
deputypc.
           ELSE.
             ls_data-deputypc = ls_mp_bp_dpc-businesspartnerfullname.
           ENDIF.
```

```
        ENDIF.
       ENDIF.
*     Get deputy project manager name
       IF ls_proj_sub-deputypm IS NOT INITIAL.
        READ TABLE lt_mp_bp_dpm INTO DATA(ls_mp_bp_dpm) WITH KEY businesspartner =
ls_proj_sub-deputypm.
        IF sy-subrc EQ 0.
         IF ls_data-deputypm IS NOT INITIAL AND ls_data-deputypm NS ls_mp_bp_dpm-
businesspartnerfullname.
          CONCATENATE ls_data-deputypm ';' ls_mp_bp_dpm-businesspartnerfullname INTO ls_data-
deputypm.
         ELSE.
          ls_data-deputypm = ls_mp_bp_dpm-businesspartnerfullname.
         ENDIF.
        ENDIF.
       ENDIF.
     ENDLOOP.
```

```
    lr_projectmanager = VALUE #( FOR wa_proj IN lt_proj ( sign = 'I' option = 'EQ'
low = wa_proj-projectmanager ) ).
    SORT lr_projectmanager BY low.
    DELETE ADJACENT DUPLICATES FROM lr_projectmanager COMPARING low.
    SELECT DISTINCT businesspartner,businesspartnerfullname FROM
i_mstrprojbpcontact INTO TABLE @DATA(lt_mp_bp_pm)
    WHERE businesspartnerrole = @lc_bprole AND businesspartner IN
@lr_projectmanager.
*    Project Controller Description
    lr_projectcontroller = VALUE #( FOR wa_proj IN lt_proj ( sign = 'I' option = 'EQ'
low = wa_proj-projectcontroller ) ).
    SORT lr_projectcontroller BY low.
    DELETE ADJACENT DUPLICATES FROM lr_projectcontroller COMPARING low.
    SELECT DISTINCT businesspartner,businesspartnerfullname FROM
i_mstrprojbpcontact INTO TABLE @DATA(lt_mp_bp_pc)
    WHERE businesspartnerrole = @lc_bprole AND businesspartner IN
@lr_projectcontroller.
*    Deputy Project Controller Description
    lr_deputypc = VALUE #( FOR wa_proj IN lt_proj ( sign = 'I' option = 'EQ' low =
wa_proj-deputypc ) ).
    LOOP AT lt_deputymc INTO DATA(ls_mc) WHERE teamrole = lc_roleid_z003.
      SPLIT ls_mc-employmentinternalids AT lc_separator INTO TABLE
DATA(lt_empids).
      LOOP AT lt_empids INTO DATA(ls_empid).
        APPEND VALUE #( sign = 'I' option = 'EQ' low = ls_empid ) TO lr_deputypc.
      ENDLOOP.
    ENDLOOP.
    SORT lr_deputypc BY low.
    DELETE ADJACENT DUPLICATES FROM lr_deputypc COMPARING low.
    SELECT DISTINCT businesspartner,businesspartnerfullname FROM
i_mstrprojbpcontact INTO TABLE @DATA(lt_mp_bp_dpc)
    WHERE businesspartnerrole = @lc_bprole AND businesspartner IN
@lr_deputypc.
*    Deputy Project Manager Description
    lr_deputypm = VALUE #( FOR wa_proj IN lt_proj ( sign = 'I' option = 'EQ' low =
wa_proj-deputypm ) ).
    LOOP AT lt_deputymc INTO ls_mc WHERE teamrole = lc_roleid_z004.
      SPLIT ls_mc-employmentinternalids AT lc_separator INTO TABLE lt_empids.
      LOOP AT lt_empids INTO ls_empid.
        APPEND VALUE #( sign = 'I' option = 'EQ' low = ls_empid ) TO lr_deputypm.
      ENDLOOP.
    ENDLOOP.
    SORT lr_deputypm BY low.
    DELETE ADJACENT DUPLICATES FROM lr_deputypm COMPARING low.
    SELECT DISTINCT businesspartner,businesspartnerfullname FROM
i_mstrprojbpcontact INTO TABLE @DATA(lt_mp_bp_dpm)
    WHERE businesspartnerrole = @lc_bprole AND businesspartner IN
@lr_deputypm.
```

```
 METHOD /iwbep/if_mgw_appl_srv_runtime~get_expanded_entityset.
**TRY.
*CALL METHOD SUPER->/IWBEP/IF_MGW_APPL_SRV_RUNTIME~GET_EXPANDED_ENTITYSET
**  EXPORTING
**    iv_entity_name          =
**    iv_entity_set_name      =
**    iv_source_name          =
**    it_filter_select_options =
**    it_order                =
**    is_paging               =
**    it_navigation_path      =
**    it_key_tab              =
**    iv_filter_string        =
**    iv_search_string        =
**    io_expand               =
**    io_tech_request_context  =
**  IMPORTING
**    er_entityset            =
**    et_expanded_clauses     =
**    et_expanded_tech_clauses =
**    es_response_context     =
*    .
**  CATCH /iwbep/cx_mgw_busi_exception.
**  CATCH /iwbep/cx_mgw_tech_exception.
**ENDTRY.

   TYPES: ty_t_itemset     TYPE TABLE OF zcl_zcon_odata_fm_fm_s_mpc=>ts_items     WITH DEFAULT
KEY,
          ty_t_plancostset TYPE TABLE OF zcl_zcon_odata_fm_fm_s_mpc=>ts_plancost  WITH DEFAULT
KEY,
          ty_t_budget      TYPE TABLE OF zcl_zcon_odata_fm_fm_s_mpc=>ts_budget_tr WITH DEFAULT
KEY,
          ty_t_actual      TYPE TABLE OF zcl_zcon_odata_fm_fm_s_mpc=>ts_actual    WITH DEFAULT
KEY,
          ty_t_pi          TYPE TABLE OF zcl_zcon_odata_fm_fm_s_mpc=>ts_piview    WITH DEFAULT KEY,
          ty_t_commit      TYPE TABLE OF zcl_zcon_odata_fm_fm_s_mpc=>ts_commit    WITH DEFAULT
KEY.
   TYPES: BEGIN OF ty_entity.
            INCLUDE TYPE zcl_zcon_odata_fm_fm_s_mpc=>ts_zcont_s_fm_sumrpt.
   TYPES:   itemsset      TYPE ty_t_itemset,
            plancostset   TYPE ty_t_plancostset,
            plancostpiset TYPE ty_t_plancostset,
            budget_trset  TYPE ty_t_budget,
            budget_relset TYPE ty_t_budget,
            actualset     TYPE ty_t_actual,
            actualpiset   TYPE ty_t_actual,
            commitset     TYPE ty_t_commit,
            piviewset     TYPE ty_t_pi,
          END OF ty_entity.

   DATA: lr_posid          TYPE /iwbep/t_cod_select_options,
```

```
      lr_kostl         TYPE /iwbep/t_cod_select_options,
      lr_bukrs         TYPE /iwbep/t_cod_select_options,
      lr_pspid         TYPE /iwbep/t_cod_select_options,
      lr_fictr         TYPE /iwbep/t_cod_select_options,
      lr_gjahr         TYPE /iwbep/t_cod_select_options,
      lr_comit         TYPE /iwbep/t_cod_select_options,
      lr_fipos         TYPE /iwbep/t_cod_select_options,
      lr_khinr         TYPE /iwbep/t_cod_select_options,
      lr_ddplan        TYPE /iwbep/t_cod_select_options,
      lr_zzplanfield   TYPE /iwbep/t_cod_select_options,
      lr_zzsubplanfield  TYPE /iwbep/t_cod_select_options,
      lr_zztechcenter  TYPE /iwbep/t_cod_select_options,
      lr_ztargetstate  TYPE /iwbep/t_cod_select_options,
      lr_zzsubsystem   TYPE /iwbep/t_cod_select_options,
      lr_zzextcooper   TYPE /iwbep/t_cod_select_options,
      lr_zzcooperdetail  TYPE /iwbep/t_cod_select_options,
      lr_vernr         TYPE /iwbep/t_cod_select_options,
      lr_parnr         TYPE /iwbep/t_cod_select_options,
      lr_stat          TYPE /iwbep/t_cod_select_options,
      lr_stage         TYPE /iwbep/t_cod_select_options,
      lrs_ddplan       LIKE LINE OF lr_ddplan,
      lr_budget_tr     TYPE /iwbep/t_cod_select_options,
      lrs_budget_tr    LIKE LINE OF lr_budget_tr,
      lr_budget_rel    TYPE /iwbep/t_cod_select_options,
      lrs_budget_rel   LIKE LINE OF lr_budget_rel,
      lr_actual        TYPE /iwbep/t_cod_select_options,
      lrs_actual       LIKE LINE OF lr_budget_rel,
      lr_commit        TYPE /iwbep/t_cod_select_options,
      lrs_commit       LIKE LINE OF lr_budget_rel,
*       ls_entity   TYPE zcl_zcon_odata_fm_fm_s_mpc=>ts_zcont_s_fm_sumrpt,
      ls_entity        TYPE ty_entity,
*       lt_entity   TYPE zcl_zcon_odata_fm_fm_s_mpc=>tt_zcont_s_fm_sumrpt,
      lt_entity        TYPE TABLE OF ty_entity,
      lt_sort          TYPE abap_sortorder_tab,
      lv_field         TYPE string,
      lv_no_result     TYPE boolean,
      lrs_bukrs        LIKE LINE OF lr_bukrs,
      lt_secid         TYPE ztfm_secid_t,
      lt_item          TYPE ztfm_budget_t,
      lt_pi           TYPE ztfm_budget_pi_t,
      lt_item_tmp      TYPE ztfm_budget_t,
      lv_monat         TYPE numc2,
      lv_bukrs         TYPE bukrs,
      lv_pspid_in      TYPE ps_pspid,
      lv_pspid_out     TYPE ps_pspid,
      ls_act_in        TYPE zfm_actual_in_s,
      lt_act_out       TYPE zfm_actual_out_t,
      lt_act_out_pi    TYPE zfm_actual_out_t,
      ls_com_in        TYPE zfm_commit_in_s,
      lt_com_out       TYPE zfm_commit_out_t,
      ls_budget_in     TYPE zfm_bgtdoc_in_s,
      lt_budget_out    TYPE zfm_bgtdoc_out_t,
      lt_budget_out_pi  TYPE zfm_bgtdoc_out_t,
      ls_plancost_in   TYPE zfm_plancost_cond_s,
      lt_plancost_out  TYPE zfm_plancost_out_cpm_t,
      lt_plancost_out_pi TYPE zfm_plancost_out_cpm_t,
      lt_restype       TYPE ztfm_budget_t,
```

```
    lt_restype_pi     TYPE ztfm_budget_pi_t,
    lt_commit         TYPE zfm_commit_out_t,
    lt_commit_pi      TYPE zfm_commit_out_t,
    lt_actual         TYPE zfm_actual_out_t,
    lt_actual_pi      TYPE zfm_actual_out_t,
    lt_bgt_tr         TYPE zfm_bgtdoc_out_t,
    lt_bgt_tr_pi      TYPE zfm_bgtdoc_out_t,
    lt_bgt_rel        TYPE zfm_bgtdoc_out_t,
    lt_bgt_rel_pi     TYPE zfm_bgtdoc_out_t,
    lt_plancost       TYPE zfm_plancost_out_cpm_t,
    lt_plancost_pi    TYPE zfm_plancost_out_cpm_t,
    lt_head           TYPE zfm_sum_rpt_head_t.

FIELD-SYMBOLS: <fs_tmp> TYPE ztfm_budget,
           <fs_set> TYPE ty_entity.

IF NOT iv_entity_set_name  = 'ZCONT_S_FM_SUMRPTSet'.
 RETURN.
ENDIF.

DATA lo_filter TYPE  REF TO /iwbep/if_mgw_req_filter.
DATA lt_filter_select_options TYPE /iwbep/t_mgw_select_option.
lo_filter = io_tech_request_context->get_filter( ).
lt_filter_select_options = lo_filter->get_filter_select_options( ).

LOOP AT it_filter_select_options ASSIGNING FIELD-SYMBOL(<fs_opt>).
 CASE <fs_opt>-property.
  WHEN 'Bukrs'.
   lr_bukrs = <fs_opt>-select_options.
   lrs_bukrs = lr_bukrs[ 1 ].
   lv_bukrs = lrs_bukrs-low.
  WHEN 'Posid'.
   lr_posid = <fs_opt>-select_options.
  WHEN 'Kostl'.
   lr_kostl = <fs_opt>-select_options.
  WHEN 'Pspid'.
   lr_pspid = <fs_opt>-select_options.
   LOOP AT lr_pspid ASSIGNING FIELD-SYMBOL(<fs_pspid>).
    lv_pspid_in =  <fs_pspid>-low.
    CALL FUNCTION 'CONVERSION_EXIT_ABPSN_INPUT'
     EXPORTING
      input  = lv_pspid_in
     IMPORTING
      output = lv_pspid_out.
    <fs_pspid>-low = lv_pspid_out.
   ENDLOOP.
  WHEN 'Fictr'.
   lr_fictr = <fs_opt>-select_options.
  WHEN 'Gjahr'.
   lr_gjahr = <fs_opt>-select_options.
  WHEN 'Fipos'.
   lr_fipos = <fs_opt>-select_options.
  WHEN 'Khinr'.
   lr_khinr = <fs_opt>-select_options.
  WHEN 'DdPlancost'.
   lr_ddplan = <fs_opt>-select_options.
  WHEN 'DdBudgetRel'.
```

```
        lr_budget_rel = <fs_opt>-select_options.
      WHEN 'DdBudgetTr'.
        lr_budget_tr = <fs_opt>-select_options.
      WHEN 'DdActual'.
        lr_actual = <fs_opt>-select_options.
      WHEN 'DdCommit'.
        lr_commit = <fs_opt>-select_options.
      WHEN 'ZzplanField'.
        lr_zzplanfield = <fs_opt>-select_options.
      WHEN 'ZzcooperDetail'.
        lr_zzcooperdetail = <fs_opt>-select_options.
      WHEN 'ZzsubPlanfield'.
        lr_zzsubplanfield = <fs_opt>-select_options.
      WHEN 'ZztechCenter'.
        lr_zztechcenter = <fs_opt>-select_options.
      WHEN 'ZzsubSystem'.
        lr_zzsubsystem = <fs_opt>-select_options.
      WHEN 'ZzextCooper'.
        lr_zzextcooper = <fs_opt>-select_options.
      WHEN 'Vernr'.
        lr_vernr = <fs_opt>-select_options.
      WHEN 'Parnr'.
        lr_parnr = <fs_opt>-select_options.
      WHEN 'Stat'.
        lr_stat = <fs_opt>-select_options.
      WHEN 'MpStage'.
        lr_stage = <fs_opt>-select_options.
      WHEN OTHERS.
    ENDCASE.
  ENDLOOP.

  AUTHORITY-CHECK OBJECT 'F_BKPF_BUK' ID 'BUKRS' FIELD lv_bukrs
                              ID 'ACTVT' FIELD '03'.
  IF sy-subrc <> 0.
    CONCATENATE 'No authority to company code ' lv_bukrs '. Please contact system administrator.'
INTO ls_entity-ztext.
    APPEND ls_entity TO lt_entity.
    copy_data_to_ref( EXPORTING is_data = lt_entity CHANGING cr_data = er_entityset ).
    RETURN.
  ENDIF.


  " get second level WBS
  CALL FUNCTION 'ZCON_FM_GET_SECID'
    EXPORTING
      ir_posid        = lr_posid
      ir_kostl        = lr_kostl
      ir_bukrs        = lr_bukrs
      ir_pspid        = lr_pspid
      ir_khinr        = lr_khinr
      ir_zzplanfield    = lr_zzplanfield
      ir_zzsubplanfield = lr_zzsubplanfield
      ir_zztechcenter   = lr_zztechcenter
      ir_zzsubsystem    = lr_zzsubsystem
      ir_zzextcooper    = lr_zzextcooper
      ir_zzcooperdetail = lr_zzcooperdetail
      ir_vernr        = lr_vernr
```

```
    ir_parnr         = lr_parnr
  IMPORTING
    ev_no_result     = lv_no_result
  CHANGING
    c_t_secid        = lt_secid.
IF lv_no_result = abap_true.
  RETURN.
ENDIF.

" get funder center
CALL FUNCTION 'ZFM_GET_FUNDCENTER'
  EXPORTING
    it_secid     = lt_secid
    ir_bukrs     = lr_bukrs
    ir_fictr     = lr_fictr
  IMPORTING
    er_fictr     = lr_fictr
    ev_no_result = lv_no_result.
IF lv_no_result = abap_true.
  RETURN.
ENDIF.

" get main data
CALL FUNCTION 'ZCON_FM_GET_DATA_MAIN'
  EXPORTING
    ir_bukrs        = lr_bukrs
    ir_fictr        = lr_fictr
    ir_comit        = lr_fipos
    ir_gjahr        = lr_gjahr
    ir_stage        = lr_stage
  IMPORTING
    et_head         = lt_head
    et_restype      = lt_restype
    et_restype_pi   = lt_restype_pi
    et_commit       = lt_commit
    et_commit_pi    = lt_commit_pi
    et_actual       = lt_actual
    et_actual_pi    = lt_actual_pi
    et_budget_rel   = lt_bgt_rel
    et_budget_rel_pi = lt_bgt_rel_pi
    et_budget_tr    = lt_bgt_tr
    et_budget_tr_pi  = lt_bgt_tr_pi
    et_plancost     = lt_plancost
    et_plancost_pi   = lt_plancost_pi.

SORT lt_actual     ASCENDING BY bukrs fistl kngjahr knbelnr knbuzei.
SORT lt_actual_pi  ASCENDING BY bukrs fistl kngjahr knbelnr knbuzei.
SORT lt_bgt_rel    ASCENDING BY fiscyear fundsctr cmmtitem docyear docnr rpmax.
SORT lt_bgt_tr     ASCENDING BY fiscyear fundsctr cmmtitem docyear docnr rpmax.
SORT lt_plancost   ASCENDING BY gjahr monat fictr cmmtitem.
SORT lt_plancost_pi ASCENDING BY gjahr pi fictr cmmtitem.
SORT lt_commit     ASCENDING BY refbn rfpos zhldt.

SELECT SINGLE fikrs
  FROM t001
  INTO @DATA(lv_fikrs)
 WHERE bukrs = @lrs_bukrs-low.
```

```
  IF NOT lt_head IS INITIAL.
   SELECT t~*
    FROM zcont_fm_tar_val AS t INNER JOIN @lt_head AS d ON t~gjahr = d~gjahr
                                      AND t~fictr = d~fictr
                                      AND t~fipex = d~cmmtitem
    INTO TABLE @DATA(lt_target).
   IF sy-subrc = 0.
    SORT lt_target BY gjahr fictr fipex.
   ENDIF.
  ENDIF.

  LOOP AT lt_head ASSIGNING FIELD-SYMBOL(<fs_head>).
   ASSIGN ls_entity TO <fs_set>.
   <fs_set>-fikrs            = lv_fikrs.
   <fs_set>-bukrs             = lrs_bukrs-low.
   <fs_set>-gjahr            = <fs_head>-gjahr.
   <fs_set>-psphi            = <fs_head>-psphi.
   CALL FUNCTION 'CONVERSION_EXIT_KONPD_OUTPUT'
    EXPORTING
     input  = <fs_set>-psphi
    IMPORTING
     output = <fs_set>-pspid.
   <fs_set>-post1            = <fs_head>-post1.
   <fs_set>-fictr            = <fs_head>-fictr.
   <fs_set>-bezeich          = <fs_head>-bezeich.
   <fs_set>-fipex            = <fs_head>-cmmtitem.
   <fs_set>-bezei            = <fs_head>-cmmtitem_txt.
   <fs_set>-display          = <fs_head>-display.
   <fs_set>-pstrt            = <fs_head>-pstrt.
   <fs_set>-pende            = <fs_head>-pende.
   <fs_set>-zpo_apv_amt      = <fs_head>-zpo_apv_amt.
   READ TABLE lt_target INTO DATA(ls_target) WITH KEY gjahr = <fs_head>-gjahr
                                      fictr = <fs_head>-fictr
                                      fipex = <fs_head>-cmmtitem.
   IF sy-subrc = 0.
    <fs_set>-ztarget = ls_target-ztarget.
   ENDIF.
   <fs_set>-wtgxxx           = <fs_head>-plancost.
   <fs_set>-ztrans_budget    = <fs_head>-budget_tr.
   <fs_set>-zrelease_budget  = <fs_head>-budget_rel.
   <fs_set>-fkbtr            = <fs_head>-commit.
   <fs_set>-zactual          = <fs_head>-actual.
*    <fs_set>-zconsumed        = <fs_head>-commit + <fs_head>-actual.
   <fs_set>-zconsumed        = <fs_head>-zpo_apv_amt + <fs_head>-actual.
*    <fs_set>-zavail           = <fs_head>-budget_rel - <fs_head>-commit - <fs_head>-actual.
   <fs_set>-zavail           = <fs_head>-budget_rel - <fs_head>-zpo_apv_amt - <fs_head>-actual.

   READ TABLE lt_restype TRANSPORTING NO FIELDS WITH KEY gjahr   = <fs_head>-gjahr
                                      fictr   = <fs_head>-fictr
                                      cmmtitem = <fs_head>-cmmtitem.
   IF sy-subrc = 0.
    DATA(lv_row) = sy-tabix.
    LOOP AT lt_restype ASSIGNING FIELD-SYMBOL(<fs_restype>) FROM lv_row WHERE gjahr   =
<fs_head>-gjahr
                                      AND fictr   = <fs_head>-fictr
                                      AND cmmtitem = <fs_head>-cmmtitem.
```

```
      DATA(lv_tabix) = sy-tabix.
      APPEND <fs_restype> TO <fs_set>-itemsset.
      DELETE lt_restype INDEX lv_tabix.
     ENDLOOP.
    ENDIF.
    READ TABLE lt_restype_pi TRANSPORTING NO FIELDS WITH KEY  gjahr    = <fs_head>-gjahr
                                        fictr    = <fs_head>-fictr
                                        cmmtitem = <fs_head>-cmmtitem.
    IF sy-subrc = 0.
     lv_row = sy-tabix.
     LOOP AT lt_restype_pi ASSIGNING FIELD-SYMBOL(<fs_res_pi>) FROM lv_row WHERE gjahr    =
<fs_head>-gjahr
                                          AND fictr    = <fs_head>-fictr
                                          AND cmmtitem = <fs_head>-cmmtitem.
      lv_tabix = sy-tabix.
      APPEND <fs_res_pi> TO <fs_set>-piviewset.
      DELETE lt_restype_pi INDEX lv_tabix.
     ENDLOOP.
    ENDIF.
    READ TABLE lt_plancost TRANSPORTING NO FIELDS WITH KEY  gjahr    = <fs_head>-gjahr
                                        fictr    = <fs_head>-fictr
                                        cmmtitem = <fs_head>-cmmtitem.
    IF sy-subrc = 0.
     lv_row = sy-tabix.
     LOOP AT lt_plancost ASSIGNING FIELD-SYMBOL(<fs_plan>) FROM lv_row WHERE gjahr    =
<fs_head>-gjahr
                                        AND fictr    = <fs_head>-fictr
                                        AND cmmtitem = <fs_head>-cmmtitem.
      lv_tabix = sy-tabix.
      APPEND <fs_plan> TO <fs_set>-plancostset.
      DELETE lt_plancost INDEX lv_tabix.
     ENDLOOP.
    ENDIF.
    READ TABLE lt_plancost_pi TRANSPORTING NO FIELDS WITH KEY  gjahr    = <fs_head>-gjahr
                                       fictr    = <fs_head>-fictr
                                       cmmtitem = <fs_head>-cmmtitem.
    IF sy-subrc = 0.
     lv_row = sy-tabix.
     LOOP AT lt_plancost_pi ASSIGNING FIELD-SYMBOL(<fs_plan_pi>) FROM lv_row WHERE gjahr    =
<fs_head>-gjahr
                                         AND fictr    = <fs_head>-fictr
                                         AND cmmtitem = <fs_head>-cmmtitem.
      lv_tabix = sy-tabix.
      APPEND <fs_plan_pi> TO <fs_set>-plancostpiset.
      DELETE lt_plancost_pi INDEX lv_tabix.
     ENDLOOP.
    ENDIF.
    READ TABLE lt_bgt_tr TRANSPORTING NO FIELDS WITH KEY  fiscyear    = <fs_head>-gjahr
                                        fundsctr    = <fs_head>-fictr
                                        cmmtitem = <fs_head>-cmmtitem.
    IF sy-subrc = 0.
     lv_row = sy-tabix.
     LOOP AT lt_bgt_tr ASSIGNING FIELD-SYMBOL(<fs_tr>) FROM lv_row WHERE fiscyear =
<fs_head>-gjahr
                                    AND fundsctr = <fs_head>-fictr
                                    AND cmmtitem = <fs_head>-cmmtitem.
      lv_tabix = sy-tabix.
```

```
      APPEND <fs_tr> TO <fs_set>-budget_trset.
      DELETE lt_bgt_tr INDEX lv_tabix.
     ENDLOOP.
    ENDIF.
    READ TABLE lt_bgt_rel TRANSPORTING NO FIELDS WITH KEY  fiscyear    = <fs_head>-gjahr
                                          fundsctr    = <fs_head>-fictr
                                          cmmtitem = <fs_head>-cmmtitem.
    IF sy-subrc = 0.
     lv_row = sy-tabix.
     LOOP AT lt_bgt_rel ASSIGNING FIELD-SYMBOL(<fs_rel>) FROM lv_row WHERE fiscyear =
<fs_head>-gjahr
                                     AND fundsctr = <fs_head>-fictr
                                     AND cmmtitem = <fs_head>-cmmtitem.
      lv_tabix = sy-tabix.
      APPEND <fs_rel> TO <fs_set>-budget_relset.
      DELETE lt_bgt_rel INDEX lv_tabix.
     ENDLOOP.
    ENDIF.
    READ TABLE lt_actual TRANSPORTING NO FIELDS WITH KEY  zrpt_year    = <fs_head>-gjahr
                                    fistl    = <fs_head>-fictr
                                    fipex = <fs_head>-cmmtitem.
    IF sy-subrc = 0.
     lv_row = sy-tabix.
     LOOP AT lt_actual ASSIGNING FIELD-SYMBOL(<fs_act>) FROM lv_row WHERE zrpt_year =
<fs_head>-gjahr
                                     AND fistl     = <fs_head>-fictr
                                     AND fipex     = <fs_head>-cmmtitem.
      lv_tabix = sy-tabix.
      APPEND <fs_act> TO <fs_set>-actualset.
      DELETE lt_actual INDEX lv_tabix.
     ENDLOOP.
    ENDIF.
    READ TABLE lt_actual_pi TRANSPORTING NO FIELDS WITH KEY  zrpt_year    = <fs_head>-gjahr
                                      fistl    = <fs_head>-fictr
                                      fipex = <fs_head>-cmmtitem.
    IF sy-subrc = 0.
     lv_row = sy-tabix.
     LOOP AT lt_actual_pi ASSIGNING FIELD-SYMBOL(<fs_act_pi>) FROM lv_row WHERE zrpt_year =
<fs_head>-gjahr
                                       AND fistl     = <fs_head>-fictr
                                       AND fipex     = <fs_head>-cmmtitem.
      lv_tabix = sy-tabix.
      APPEND <fs_act_pi> TO <fs_set>-actualpiset.
      DELETE lt_actual_pi INDEX lv_tabix.
     ENDLOOP.
    ENDIF.
    READ TABLE lt_commit TRANSPORTING NO FIELDS WITH KEY  zrpt_year = <fs_head>-gjahr
                                    fistl    = <fs_head>-fictr
                                    fipex = <fs_head>-cmmtitem.
    IF sy-subrc = 0.
     lv_row = sy-tabix.
     LOOP AT lt_commit ASSIGNING FIELD-SYMBOL(<fs_com>) FROM lv_row WHERE zrpt_year =
<fs_head>-gjahr
                                     AND fistl     = <fs_head>-fictr
                                     AND fipex     = <fs_head>-cmmtitem.
      lv_tabix = sy-tabix.
      APPEND <fs_com> TO <fs_set>-commitset.
```

```
      DELETE lt_commit INDEX lv_tabix.
    ENDLOOP.
  ENDIF.
  APPEND ls_entity TO lt_entity.
  CLEAR: ls_entity.
 ENDLOOP.

 SORT lt_entity BY gjahr fictr fipex.
 IF io_tech_request_context->has_inlinecount( ) = abap_true.
  DESCRIBE TABLE lt_entity LINES es_response_context-inlinecount.
 ELSE.
  CLEAR es_response_context-inlinecount.
 ENDIF.

 copy_data_to_ref( EXPORTING is_data = lt_entity CHANGING cr_data = er_entityset ).

ENDMETHOD.
```

```
FUNCTION zfm_plancost.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(IS_INPUT) TYPE  ZTFM_COND01_S
*"     REFERENCE(IT_GJAHR) TYPE  BSPL_GJAHR_T
*"     REFERENCE(IT_DATA) TYPE  ZTFM_BUDGET_T
*"  CHANGING
*"     REFERENCE(CT_PLANCOST) TYPE  ZFM_PLANCOST_OUT_CPM_T
*"     REFERENCE(CT_PLANCOST_PI) TYPE  ZFM_PLANCOST_OUT_CPM_T
*"----------------------------------------------------------------------

 TYPES: BEGIN OF ty_plancost,
         fictr TYPE fistl,
         posnr TYPE ps_posnr,
         fipex TYPE fipex,
        END OF ty_plancost.

 DATA: lv_posid    TYPE prps-posid,
       lv_posnr    TYPE ps_posnr,
       lv_monat    TYPE monat VALUE '00',
       lt_keyfig   TYPE rsd_t_dta_pro,
       lt_char     TYPE rsd_t_dta_pro,
       lt_message  TYPE bsanly_t_message,
       ls_range    TYPE rsdri_s_range,
       lt_range    TYPE rsdri_t_range,
       lr_data     TYPE REF TO data,
       lv_plancost TYPE wrbtr,
       lv_plan_qty TYPE menge_d,
       lv_fictr    TYPE fistl,
       lv_gjahr    TYPE gjahr,
       lv_last     TYPE sy-datum,
       lv_cost     TYPE wrbtr,
       lv_left     TYPE wrbtr,
       lv_fipex    TYPE fipex,
       lr_posnr    TYPE RANGE OF ps_posnr,
       lt_pr       TYPE TABLE OF zptpt_prepr,
       lt_plancost TYPE zfm_plancost_out_cpm_t.


 CALL METHOD /cpd/cl_pfp_ip_service=>get_cube_keyfig_name
   EXPORTING
    iv_infoprov = /cpd/cl_pfp_constants=>gc_pfp_cube
   IMPORTING
    et_dta_pro  = lt_keyfig.

 CALL METHOD /cpd/cl_pfp_ip_service=>get_cube_char_name
   EXPORTING
    iv_infoprov = /cpd/cl_pfp_constants=>gc_pfp_cube
   IMPORTING
    et_dta_pro  = lt_char.
```

```
* Retain the Required Key figures
 DELETE lt_keyfig WHERE iobjnm <> /cpd/cl_pfp_constants=>gc_tcost
                  AND iobjnm <> /cpd/cl_pfp_constants=>gc_ftrate
                  AND iobjnm <> /cpd/cl_pfp_constants=>gc_qty
                  AND iobjnm <> /cpd/cl_pfp_constants=>gc_pcost.
* Retain the Required char
 DELETE lt_char WHERE iobjnm <> /cpd/cl_pfp_constants=>gc_fpoid
                AND iobjnm <> /cpd/cl_pfp_constants=>gc_fver
                AND iobjnm <> /cpd/cl_pfp_constants=>gc_fpid
                AND iobjnm <> /cpd/cl_pfp_constants=>gc_frtyp
                AND iobjnm <> /cpd/cl_pfp_constants=>gc_calmonth.

 DATA(lt_data) = it_data.
 LOOP AT lt_data ASSIGNING FIELD-SYMBOL(<fs_data>).
  CALL FUNCTION 'CONVERSION_EXIT_ABPSP_INPUT'
    EXPORTING
     input     = <fs_data>-fictr
    IMPORTING
     output    = <fs_data>-posid
    EXCEPTIONS
     not_found = 1
     OTHERS    = 2.
 ENDLOOP.

 DATA(lv_bukrs) = is_input-bukrs[ 1 ]-low.

 SELECT SINGLE fikrs
  FROM t001
  INTO @DATA(lv_fikrs)
 WHERE bukrs = @lv_bukrs.

 SELECT r~*
  FROM zptpt_prepr AS r INNER JOIN @lt_data AS t ON r~ps_psp_pnr = t~posid
*   WHERE ps_psp_pnr    IN @lr_posnr
  WHERE controlstatus EQ 'A'
    AND preprsta      NE 'K'
  INTO TABLE @DATA(lt_prepr)    .
 IF sy-subrc = 0.
  SORT lt_prepr BY prepurreqn prebnfpo.
  DELETE ADJACENT DUPLICATES FROM lt_prepr COMPARING prepurreqn prebnfpo.
  SORT lt_prepr BY ps_psp_pnr.
 ENDIF.
 SELECT *
  FROM fmfctrt
  INTO TABLE @DATA(lt_fmfctrt)
 WHERE fictr IN @is_input-fictr
   AND spras EQ @sy-langu.
 IF sy-subrc = 0.
  SORT lt_fmfctrt BY fictr fikrs spras.
 ENDIF.
 SELECT *
  FROM fmcit
  INTO TABLE @DATA(lt_fmcit)
 WHERE fipex IN @is_input-cmmtitem
   AND spras EQ @sy-langu.
 IF sy-subrc = 0.
  SORT lt_fmcit BY fikrs fipex.
```

```
 ENDIF.
 SELECT *
  FROM prps
  INTO TABLE @DATA(lt_prps)
 WHERE pspnr IN @lr_posnr.
IF sy-subrc = 0.
 SORT lt_prps BY pspnr.
ENDIF.
SELECT *
 FROM zconv_fm_pi_view
 INTO TABLE @DATA(lt_view)
WHERE gjahr IN @is_input-gjahr.

IF NOT lt_prps IS INITIAL.
 SELECT p~*
  FROM prhi AS p INNER JOIN @lt_prps AS t ON p~down = t~pspnr
  INTO TABLE @DATA(lt_prhi).
 IF sy-subrc = 0.
  SELECT p~*
   FROM prps AS p INNER JOIN @lt_prhi AS i ON p~pspnr = i~posnr
   INTO TABLE @DATA(lt_first).
  IF sy-subrc = 0.
   SELECT m~*
    FROM /cpd/d_mp_item AS m INNER JOIN @lt_first AS t ON m~mp_item_okey = t~objnr
    INTO TABLE @DATA(lt_mp_item).
   IF sy-subrc = 0.
    SELECT h~*
     FROM /cpd/d_pfp_ph AS h INNER JOIN @lt_mp_item AS t ON h~mp_id_int = t~parent_key
     WHERE h~sel_structure ='E'
     INTO TABLE @DATA(lt_ph)          .
    IF sy-subrc = 0.
     SELECT v~*
      FROM /cpd/d_pfp_pv AS v INNER JOIN @lt_ph AS t ON v~parent_key = t~db_key
      WHERE v~version_type = 'PLAN'
      INTO TABLE @DATA(lt_pv)           .
    ENDIF.
   ENDIF.
  ENDIF.
 ENDIF.

 SORT lt_prhi    BY down.
 SORT lt_first   BY pspnr.
 SORT lt_mp_item BY mp_item_okey.
 SORT lt_ph      BY mp_id_int.


 LOOP AT lt_data ASSIGNING <fs_data>.
  READ TABLE lt_prhi ASSIGNING FIELD-SYMBOL(<fs_prhi>) WITH KEY down = <fs_data>-posid
BINARY SEARCH.
  IF sy-subrc = 0.
   READ TABLE lt_first ASSIGNING FIELD-SYMBOL(<fs_first>) WITH KEY pspnr = <fs_prhi>-posnr
BINARY SEARCH.
   IF sy-subrc = 0.
    READ TABLE lt_mp_item ASSIGNING FIELD-SYMBOL(<fs_item>) WITH KEY mp_item_okey =
<fs_first>-objnr BINARY SEARCH.
    IF sy-subrc = 0.
```

```
      READ TABLE lt_ph ASSIGNING FIELD-SYMBOL(<fs_ph>) WITH KEY mp_id_int     = <fs_item>-
parent_key
                                          sel_structure = 'E' BINARY SEARCH.
      IF sy-subrc = 0.
        IF <fs_ph>-zzplan_option = 'BU' AND ( <fs_data>-cmmtitem = 'EXTERNAL' OR <fs_data>-
cmmtitem = 'ASSET' ).
          READ TABLE lt_prepr TRANSPORTING NO FIELDS WITH KEY ps_psp_pnr = <fs_data>-posid.
          IF sy-subrc = 0.
            DATA(lv_tabix) = sy-tabix.
            LOOP AT lt_prepr ASSIGNING FIELD-SYMBOL(<fs_prepr>) FROM lv_tabix WHERE
ps_psp_pnr = <fs_data>-posid.
              IF <fs_prepr>-lfdat IS INITIAL.
                <fs_prepr>-lfdat = <fs_prepr>-startdate.
                <fs_prepr>-prepurreqnitmdescription = <fs_data>-fictr.
                APPEND <fs_prepr> TO lt_pr.
              ENDIF.
            ENDLOOP.
          ENDIF.
          " monthly view
          CLEAR: lv_monat.
          DO 12 TIMES.
            lv_monat = lv_monat + 1.
            APPEND INITIAL LINE TO ct_plancost ASSIGNING FIELD-SYMBOL(<fs_output>).
            <fs_output>-gjahr       = <fs_data>-gjahr.
            <fs_output>-monat       = lv_monat.
            <fs_output>-posid       = <fs_data>-fictr.
            READ TABLE lt_prps ASSIGNING FIELD-SYMBOL(<fs_prps>) WITH KEY pspnr = <fs_data>-
posid BINARY SEARCH.
            IF sy-subrc = 0.
              <fs_output>-post1     = <fs_prps>-post1.
            ENDIF.
            <fs_output>-fictr       = <fs_data>-fictr.
            READ TABLE lt_fmfctrt ASSIGNING FIELD-SYMBOL(<fs_fmfctrt>) WITH KEY fictr = <fs_data>-
fictr
                                                fikrs = lv_fikrs
                                                spras = sy-langu BINARY SEARCH.
            IF sy-subrc = 0.
              <fs_output>-bezeich    = <fs_fmfctrt>-beschr.
            ENDIF.
            <fs_output>-cmmtitem     = <fs_data>-cmmtitem.
            READ TABLE lt_fmcit ASSIGNING FIELD-SYMBOL(<fs_fmcit>) WITH KEY fikrs = lv_fikrs
                                                  fipex = <fs_data>-cmmtitem BINARY SEARCH.
            IF sy-subrc = 0.
              <fs_output>-cmmtitem_txt = <fs_fmcit>-bezei.
            ENDIF.
            READ TABLE lt_prepr TRANSPORTING NO FIELDS WITH KEY ps_psp_pnr = <fs_data>-posid.
            IF sy-subrc = 0.
              lv_tabix = sy-tabix.
              LOOP AT lt_prepr ASSIGNING <fs_prepr> FROM lv_tabix WHERE lfdat(6) = <fs_data>-gjahr
&& lv_monat
                                      AND ps_psp_pnr = <fs_data>-posid.
                IF <fs_data>-cmmtitem = 'ASSET' AND <fs_prepr>-assetflag IS INITIAL.
                  CONTINUE.
                ENDIF.
                IF <fs_data>-cmmtitem = 'EXTERNAL' AND <fs_prepr>-assetflag IS NOT INITIAL.
                  CONTINUE.
                ENDIF.
```

```
            <fs_output>-plancost = <fs_output>-plancost + <fs_prepr>-localplanvalue.
           ENDLOOP.
          ENDIF.
         ENDDO.
        ELSE.
         ls_range-chanm  = /cpd/cl_pfp_constants=>gc_fpoid. "Name Of Characteristic
         ls_range-sign   = 'I'.              "Include
         ls_range-compop = 'EQ'.             "Operator
         ls_range-low    = <fs_ph>-plan_id.
         APPEND ls_range TO lt_range.
         CLEAR ls_range.
         READ TABLE lt_pv ASSIGNING FIELD-SYMBOL(<fs_pv>) WITH KEY parent_key   = <fs_ph>-
db_key
                                                   version_type = 'PLAN'.
         IF sy-subrc = 0.
          ls_range-chanm  = /cpd/cl_pfp_constants=>gc_fver. "Name Of Characteristic
          ls_range-sign   = 'I'.              "Include
          ls_range-compop = 'EQ'.             "Operator
          ls_range-low    = <fs_pv>-version_id.
          APPEND ls_range TO lt_range.
         ENDIF.
*             "Pass filter for CALMONTH
         CLEAR ls_range.
         ls_range-chanm  = /cpd/cl_pfp_constants=>gc_calmonth. "Name Of Characteristic
         ls_range-sign   = 'I'.              "Include
         ls_range-compop = 'BT'.
         ls_range-low    = <fs_data>-gjahr && '01'.
         ls_range-high   = <fs_data>-gjahr && '12'.
         APPEND ls_range TO lt_range.
         CLEAR ls_range.
         ls_range-chanm  = /cpd/cl_pfp_constants=>gc_frtyp. "Resource Type
         ls_range-sign   = 'I'.              "Include
         ls_range-compop = 'EQ'.             "Operator
         IF <fs_data>-cmmtitem = 'INTERNAL'.
          ls_range-low = 'ZINT'.
         ELSEIF <fs_data>-cmmtitem = 'EXTERNAL'.
          ls_range-low = 'ZEXT'.
         ELSE.
          ls_range-low = 'ZAST'.
         ENDIF.
         APPEND ls_range TO lt_range.
         CLEAR ls_range.
         ls_range-chanm  = /cpd/cl_pfp_constants=>gc_fpid . "WBS
         ls_range-sign   = 'I'.              "Include
         ls_range-compop = 'EQ'.             "Operator
         ls_range-low    = 'PR' && <fs_data>-posid.
         APPEND ls_range TO lt_range.
        ENDIF.
       ENDIF.
      ENDIF.
     ENDIF.
    ENDIF.
   ENDLOOP.
   SORT lt_range BY chanm sign compop low high.
   DELETE ADJACENT DUPLICATES FROM lt_range COMPARING chanm sign compop low high.
*** Read data from infocube.
   CALL METHOD /cpd/cl_pfp_query_services=>get_infocube_data
```

```
  EXPORTING
    it_keyfigure = lt_keyfig
    it_char      = lt_char
    it_range     = lt_range
  IMPORTING
    er_t_data    = lr_data
    et_message   = lt_message.
ASSIGN lr_data->* TO FIELD-SYMBOL(<ft_data>).
IF <ft_data> IS ASSIGNED.
  LOOP AT <ft_data> ASSIGNING FIELD-SYMBOL(<fs_data1>).
    CLEAR: lv_plancost,
           lv_plan_qty,
           lv_fipex,
           lv_gjahr,
           lv_monat,
           lv_posnr,
           lv_fictr.
    ASSIGN COMPONENT '/CPD/FPCA' OF STRUCTURE <fs_data1> TO FIELD-SYMBOL(<fs_fpca>).
    lv_plancost = <fs_fpca>.
    ASSIGN COMPONENT '/CPD/FQTY' OF STRUCTURE <fs_data1> TO FIELD-SYMBOL(<fs_fqty>).
    lv_plan_qty  = <fs_fqty>.
    ASSIGN COMPONENT '/CPD/FRTYP' OF STRUCTURE <fs_data1> TO FIELD-SYMBOL(<fs_fipex>).
    IF <fs_fipex> = 'ZEXT'.
      lv_fipex = 'EXTERNAL'.
    ELSEIF  <fs_fipex> = 'ZINT'.
      lv_fipex = 'INTERNAL'.
    ELSE.
      lv_fipex = 'ASSET'.
    ENDIF.
    ASSIGN COMPONENT '0CALMONTH' OF STRUCTURE <fs_data1> TO FIELD-SYMBOL(<fs_month>).
    lv_gjahr = <fs_month>(4).
    lv_monat = <fs_month>+4(2).
    ASSIGN COMPONENT '/CPD/FPID' OF STRUCTURE <fs_data1> TO FIELD-SYMBOL(<fs_fpid>).
    lv_posnr = <fs_fpid>+2.
    CALL FUNCTION 'CONVERSION_EXIT_ABPSP_OUTPUT'
      EXPORTING
        input     = lv_posnr
      IMPORTING
        output    = lv_fictr
      EXCEPTIONS
        not_found = 1
        OTHERS    = 2.
    APPEND INITIAL LINE TO lt_plancost ASSIGNING <fs_output>.
    <fs_output>-fictr    = lv_fictr.
    READ TABLE lt_fmfctrt ASSIGNING <fs_fmfctrt> WITH KEY fictr = <fs_output>-fictr
                                        fikrs = lv_fikrs
                                        spras = sy-langu.
    IF sy-subrc = 0.
      <fs_output>-bezeich = <fs_fmfctrt>-beschr.
    ENDIF.
    <fs_output>-posid    = lv_fictr.
    READ TABLE lt_prps ASSIGNING <fs_prps> WITH KEY pspnr = <fs_output>-posid.
    IF sy-subrc = 0.
      <fs_output>-post1     = <fs_prps>-post1.
    ENDIF.
    <fs_output>-gjahr    = lv_gjahr.
    <fs_output>-monat    = lv_monat.
```

```
    <fs_output>-cmmtitem = lv_fipex.
    READ TABLE lt_fmcit ASSIGNING <fs_fmcit> WITH KEY fikrs = lv_fikrs
                                      fipex = <fs_output>-cmmtitem.
    IF sy-subrc = 0.
      <fs_output>-cmmtitem_txt = <fs_fmcit>-bezei.
    ENDIF.
    <fs_output>-plancost  = lv_plancost.
    <fs_output>-zplan_qty = lv_plan_qty.
  ENDLOOP.
  UNASSIGN <ft_data>.
ENDIF.

DATA(lt_plan) = ct_plancost.


LOOP AT lt_data ASSIGNING <fs_data>.
  READ TABLE lt_plan TRANSPORTING NO FIELDS WITH KEY fictr    = <fs_data>-fictr
                                     gjahr    = <fs_data>-gjahr
                                     cmmtitem = <fs_data>-cmmtitem.
  IF sy-subrc <> 0.
    READ TABLE lt_plancost TRANSPORTING NO FIELDS WITH KEY fictr    = <fs_data>-fictr
                                       gjahr    = <fs_data>-gjahr
                                       cmmtitem = <fs_data>-cmmtitem.
    IF sy-subrc = 0.
      CLEAR: lv_monat.
      DO 12 TIMES.
        lv_monat = lv_monat + 1.
        READ TABLE lt_plancost INTO DATA(ls_plan) WITH KEY monat    = lv_monat
                                       fictr    = <fs_data>-fictr
                                       gjahr    = <fs_data>-gjahr
                                       cmmtitem = <fs_data>-cmmtitem.
        IF sy-subrc = 0.
          ls_plan-posid = <fs_data>-fictr.
          ls_plan-post1 = <fs_data>-bezeich.
          APPEND ls_plan TO ct_plancost.
        ELSE.
          MOVE-CORRESPONDING <fs_data> TO ls_plan.
          ls_plan-monat = lv_monat.
          ls_plan-posid = ls_plan-fictr.
          ls_plan-post1 = ls_plan-bezeich.
          APPEND ls_plan TO ct_plancost.
        ENDIF.
      ENDDO.
    ELSE.
      CLEAR: lv_monat.
      DO 12 TIMES.
        lv_monat = lv_monat + 1.
        MOVE-CORRESPONDING <fs_data> TO ls_plan.
        ls_plan-monat = lv_monat.
        ls_plan-posid = ls_plan-fictr.
        READ TABLE lt_fmfctrt ASSIGNING <fs_fmfctrt> WITH KEY fictr = ls_plan-fictr
                                         fikrs = lv_fikrs
                                         spras = sy-langu.
        IF sy-subrc = 0.
          ls_plan-bezeich = <fs_fmfctrt>-bezeich.
          ls_plan-post1   = <fs_fmfctrt>-bezeich.
        ENDIF.
```

```
        APPEND ls_plan TO ct_plancost.
       ENDDO.
      ENDIF.
     ENDIF.

    ENDLOOP.

    SORT ct_plancost ASCENDING BY gjahr fictr cmmtitem monat.
    " PI View
    LOOP AT ct_plancost ASSIGNING FIELD-SYMBOL(<fs_cost>) GROUP BY ( gjahr    = <fs_cost>-gjahr
                                                                     fictr    = <fs_cost>-fictr
                                                                     cmmtitem = <fs_cost>-cmmtitem ).
      LOOP AT lt_view ASSIGNING FIELD-SYMBOL(<fs_view>).
       APPEND INITIAL LINE TO ct_plancost_pi ASSIGNING FIELD-SYMBOL(<fs_pi>).
       <fs_pi>-gjahr       = <fs_cost>-gjahr.
       <fs_pi>-pi          = <fs_view>-pi.
       <fs_pi>-posid       = <fs_cost>-fictr.
       <fs_pi>-post1       = <fs_cost>-post1.
       <fs_pi>-fictr       = <fs_cost>-fictr.
       <fs_pi>-bezeich     = <fs_cost>-bezeich.
       <fs_pi>-cmmtitem    = <fs_cost>-cmmtitem.
       <fs_pi>-cmmtitem_txt = <fs_cost>-cmmtitem_txt.
       IF <fs_cost>-cmmtitem = 'EXTERNAL'.
        READ TABLE lt_pr TRANSPORTING NO FIELDS WITH KEY gjahr = <fs_cost>-gjahr
prepurreqnitmdescription = <fs_cost>-fictr.
        IF sy-subrc = 0.
         CONTINUE.
        ENDIF.
       ENDIF.
       LOOP AT GROUP <fs_cost> INTO DATA(ls_cost).
        DATA(lv_time) = ls_cost-gjahr && ls_cost-monat.
        IF lv_time >= <fs_view>-zdatefrom(6) AND lv_time <= <fs_view>-zdateto(6).
         IF lv_time = <fs_view>-zdatefrom(6) AND lv_left IS NOT INITIAL.
          CALL FUNCTION 'RP_LAST_DAY_OF_MONTHS'
           EXPORTING
            day_in          = <fs_view>-zdatefrom
           IMPORTING
            last_day_of_month = lv_last
           EXCEPTIONS
            day_in_no_date    = 1
            OTHERS            = 2.
          DATA(lv_cur) = <fs_view>-zdatefrom+6(2).
          DATA(lv_end) = lv_last+6(2).
          DATA(lv_ran) = lv_end - lv_cur.
          <fs_pi>-plancost = lv_left.
          CLEAR: lv_cur,
                 lv_end,
                 lv_ran,
                 lv_cost,
                 lv_left.
          CONTINUE.
         ENDIF.
         IF lv_time = <fs_view>-zdateto(6).
          CALL FUNCTION 'RP_LAST_DAY_OF_MONTHS'
           EXPORTING
            day_in          = <fs_view>-zdateto
           IMPORTING
```

```
            last_day_of_month = lv_last
           EXCEPTIONS
            day_in_no_date    = 1
            OTHERS            = 2.
          lv_cur = <fs_view>-zdateto+6(2).
          lv_end = lv_last+6(2).
          lv_ran = lv_end - lv_cur.
          lv_cost = ls_cost-plancost * lv_cur / lv_end.
          lv_left = ls_cost-plancost - lv_cost.
          <fs_pi>-plancost = <fs_pi>-plancost + lv_cost.
          CLEAR: lv_cur,
               lv_end,
               lv_ran,
               lv_cost.
          CONTINUE.
        ENDIF.
        <fs_pi>-plancost = <fs_pi>-plancost + ls_cost-plancost.
      ENDIF.
    ENDLOOP.
  ENDLOOP.
 ENDLOOP.

 LOOP AT lt_pr ASSIGNING <fs_prepr>.
  READ TABLE ct_plancost_pi ASSIGNING <fs_pi> WITH KEY fictr    = <fs_prepr>-
prepurreqnitmdescription
                                    gjahr    = <fs_prepr>-gjahr
                                    pi       = <fs_prepr>-pinum
                                    cmmtitem = 'EXTERNAL'.
  IF sy-subrc = 0.
   <fs_pi>-plancost = <fs_pi>-plancost + <fs_prepr>-planvalue.
  ENDIF.
 ENDLOOP.

 CLEAR: lr_data,
       lt_range,
       lt_message.


ENDFUNCTION.
```

```
FUNCTION zfm_get_budget_detail.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(IS_INPUT) TYPE  ZFM_BGTDOC_IN_S
*"  EXPORTING
*"     REFERENCE(ET_OUTPUT) TYPE  ZFM_BGTDOC_OUT_T
*"----------------------------------------------------------------------

  DATA: lv_num      TYPE numc2,
        lv_username TYPE bapibname-bapibname,
        ls_address  TYPE bapiaddr3,
        lt_return   TYPE bapiret2_t,
        lv_last     TYPE sy-datum,
        ls_output   TYPE zfm_bgtdoc_out_s.

  FIELD-SYMBOLS:
    <fs_value> TYPE any.

  SELECT *
    FROM zconv_fm_pi_view
    INTO TABLE @DATA(lt_pi)
   WHERE gjahr IN @is_input-gjahr.
  SORT lt_pi ASCENDING BY gjahr pi.

  SELECT f1~fm_area,                              "#EC CI_SUBRC
        f1~docyear,
        f1~docnr,
        f1~process_ui,
        f1~doctype,
        f1~version,
        f1~crtuser,
        f1~crtdate,
        f1~crttime,
        f1~docdate,
        f1~resppers,
        f1~text50,
        f1~docstate,
        f1~revstate,
        f1~rev_refnr,
        f1~zbgsrc,
        z1~txt40 AS bgsrc_txt,
*       f1~ddate,
        f1~zdate,
        f1~zlink,
        f2~docln,
        f2~rpmax,
        f2~fiscyear,
        f2~fundsctr,
        f2~text50 AS text50_itm,
        f3~beschr AS bezeich,
```

```
    f2~cmmtitem,
    f4~bezei AS cmmtitem_txt,
    f2~budcat,
    f2~valtype,
    f2~budtype,
    f2~tcurr,
    f2~tval01,
    f2~tval02,
    f2~tval03,
    f2~tval04,
    f2~tval05,
    f2~tval06,
    f2~tval07,
    f2~tval08,
    f2~tval09,
    f2~tval10,
    f2~tval11,
    f2~tval12,
    f2~tval13,
    f2~tval14,
    f2~tval15,
    f2~tval16
 FROM fmbh AS f1 INNER JOIN fmbl        AS f2 ON f1~fm_area = f2~fm_area
                                 AND f1~docyear = f2~docyear
                                 AND f1~docnr   = f2~docnr
        LEFT OUTER JOIN fmfctrt      AS f3 ON f3~fictr   = f2~fundsctr
                                 AND f3~spras   = @sy-langu
        LEFT OUTER JOIN fmcit        AS f4 ON f4~fipex   = f2~cmmtitem
                                 AND f4~spras   = @sy-langu
        LEFT OUTER JOIN ztfm_bgt_src AS z1 ON f1~zbgsrc  = z1~bgtsrc
 WHERE f1~fm_area  IN @is_input-fikrs
  AND f2~fundsctr IN @is_input-fictr
  AND f2~cmmtitem IN @is_input-cmmtitem
  AND f1~docyear  IN @is_input-gjahr
  AND f2~valtype  IN @is_input-valtype
  AND f1~docstate IN @is_input-docstate
  AND ( f1~docstate  = '1' OR f1~docstate  = '3' )
 INTO TABLE @DATA(lt_output).
##INTO_OK

LOOP AT lt_output INTO DATA(ls_output_tmp).

 CLEAR ls_output.
 MOVE-CORRESPONDING ls_output_tmp TO ls_output.

 " get users' full name
 lv_username = ls_output-crtuser.
 CALL FUNCTION 'BAPI_USER_GET_DETAIL'
  EXPORTING
   username = lv_username
  IMPORTING
   address  = ls_address
  TABLES
   return   = lt_return.
 IF sy-subrc  = 0.
  ls_output-ename = ls_address-fullname.
 ENDIF.
```

```
  CLEAR:lv_username,
        ls_address,
        lt_return.

  ##INTO_OK
  LOOP AT is_input-monat[] INTO DATA(ls_monat).       "#EC CI_NESTED
    DATA(lv_field) = 'LS_OUTPUT_TMP-TVAL' && ls_monat-low.
    ASSIGN (lv_field) TO <fs_value>.                  "#EC CI_SUBRC
    IF <fs_value> IS ASSIGNED AND <fs_value> IS NOT INITIAL.
      ls_output-tvalx = 0 - <fs_value>.
    ENDIF.

    lv_field = 'LS_OUTPUT_TMP-LVAL' && ls_monat-low.
    ASSIGN (lv_field) TO <fs_value>.                  "#EC CI_SUBRC
    IF <fs_value> IS ASSIGNED AND <fs_value> IS NOT INITIAL.
      ls_output-lvalx = 0 - <fs_value>.
    ENDIF.

    IF ls_output-tvalx IS NOT INITIAL OR ls_output-lvalx IS NOT INITIAL.
      ls_output-rpmax = ls_monat-low.
      APPEND ls_output TO et_output.
      CLEAR:
       ls_output-tvalx,
       ls_output-lvalx.
    ENDIF.

  ENDLOOP.

  IF is_input-monat[] IS INITIAL.
    DO 16 TIMES.                                    "#EC CI_NESTED
      lv_num = lv_num + 1.
      DATA(lv_field2) = 'LS_OUTPUT_TMP-TVAL' && lv_num.
      ASSIGN (lv_field2) TO <fs_value>.               "#EC CI_SUBRC
      IF <fs_value> IS ASSIGNED AND <fs_value> IS NOT INITIAL.
        ls_output-tvalx = 0 - <fs_value>.
      ENDIF.

      lv_field2 = 'LS_OUTPUT_TMP-LVAL' && lv_num.
      ASSIGN (lv_field2) TO <fs_value>.               "#EC CI_SUBRC
      IF <fs_value> IS ASSIGNED AND <fs_value> IS NOT INITIAL.
        ls_output-lvalx = 0 - <fs_value>.
      ENDIF.

      IF ls_output-tvalx IS NOT INITIAL OR ls_output-lvalx IS NOT INITIAL.
        ls_output-rpmax = lv_num.
        APPEND ls_output TO et_output.
        CLEAR:
          ls_output-tvalx,
          ls_output-lvalx.
      ENDIF.
    ENDDO.
    CLEAR:
      lv_num.
  ENDIF.
ENDLOOP.

LOOP AT et_output ASSIGNING FIELD-SYMBOL(<fs_budget>).
```

```
   DATA(lv_time) = <fs_budget>-fiscyear && <fs_budget>-rpmax+1(2).
   LOOP AT lt_pi ASSIGNING FIELD-SYMBOL(<fs_pi>) WHERE zdatefrom(6) <= lv_time
                                    AND zdateto(6)   >= lv_time.
     <fs_budget>-pi = <fs_pi>-pi.
     EXIT.
   ENDLOOP.
  ENDLOOP.

ENDFUNCTION.
```

```
FUNCTION zfm_get_budget_detail.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(IS_INPUT) TYPE  ZFM_BGTDOC_IN_S
*"  EXPORTING
*"     REFERENCE(ET_OUTPUT) TYPE  ZFM_BGTDOC_OUT_T
*"----------------------------------------------------------------------

  DATA: lv_num      TYPE numc2,
        lv_username TYPE bapibname-bapibname,
        ls_address  TYPE bapiaddr3,
        lt_return   TYPE bapiret2_t,
        lv_last     TYPE sy-datum,
        ls_output   TYPE zfm_bgtdoc_out_s.

  FIELD-SYMBOLS:
    <fs_value> TYPE any.

  SELECT *
    FROM zconv_fm_pi_view
    INTO TABLE @DATA(lt_pi)
   WHERE gjahr IN @is_input-gjahr.
  SORT lt_pi ASCENDING BY gjahr pi.

  SELECT f1~fm_area,                              "#EC CI_SUBRC
         f1~docyear,
         f1~docnr,
         f1~process_ui,
         f1~doctype,
         f1~version,
         f1~crtuser,
         f1~crtdate,
         f1~crttime,
         f1~docdate,
         f1~resppers,
         f1~text50,
         f1~docstate,
         f1~revstate,
         f1~rev_refnr,
         f1~zbgsrc,
         z1~txt40 AS bgsrc_txt,
*        f1~ddate,
         f1~zdate,
         f1~zlink,
         f2~docln,
         f2~rpmax,
         f2~fiscyear,
         f2~fundsctr,
         f2~text50 AS text50_itm,
         f3~beschr AS bezeich,
```

```
      f2~cmmtitem,
      f4~bezei AS cmmtitem_txt,
      f2~budcat,
      f2~valtype,
      f2~budtype,
      f2~tcurr,
      f2~tval01,
      f2~tval02,
      f2~tval03,
      f2~tval04,
      f2~tval05,
      f2~tval06,
      f2~tval07,
      f2~tval08,
      f2~tval09,
      f2~tval10,
      f2~tval11,
      f2~tval12,
      f2~tval13,
      f2~tval14,
      f2~tval15,
      f2~tval16,

      u1~userdescription as ename
  FROM fmbh AS f1 INNER JOIN fmbl        AS f2 ON f1~fm_area = f2~fm_area
                                AND f1~docyear = f2~docyear
                                AND f1~docnr   = f2~docnr
         LEFT OUTER JOIN fmfctrt      AS f3 ON f3~fictr   = f2~fundsctr
                                AND f3~spras   = @sy-langu
         LEFT OUTER JOIN fmcit        AS f4 ON f4~fipex   = f2~cmmtitem
                                AND f4~spras   = @sy-langu
         LEFT OUTER JOIN ztfm_bgt_src AS z1 ON f1~zbgsrc  = z1~bgtsrc
         LEFT OUTER JOIN i_userdescription WITH PRIVILEGED ACCESS AS u1 ON f1~crtuser  = u1
~userid

  WHERE f1~fm_area  IN @is_input-fikrs
    AND f2~fundsctr IN @is_input-fictr
    AND f2~cmmtitem IN @is_input-cmmtitem
    AND f1~docyear  IN @is_input-gjahr
    AND f2~valtype  IN @is_input-valtype
    AND f1~docstate IN @is_input-docstate
    AND ( f1~docstate  = '1' OR f1~docstate  = '3' )
   INTO TABLE @DATA(lt_output).
 ##INTO_OK
 LOOP AT lt_output INTO DATA(ls_output_tmp).

   CLEAR ls_output.
   MOVE-CORRESPONDING ls_output_tmp TO ls_output.

   " get users' full name
*    lv_username = ls_output-crtuser.
*    CALL FUNCTION 'BAPI_USER_GET_DETAIL'
*      EXPORTING
*        username = lv_username
*      IMPORTING
*        address  = ls_address
*      TABLES
```

```
*      return   = lt_return.
*    IF sy-subrc  = 0.
*      ls_output-ename = ls_address-fullname.
*    ENDIF.
*    CLEAR:lv_username,
*          ls_address,
*          lt_return.

    ##INTO_OK
    LOOP AT is_input-monat[] INTO DATA(ls_monat).       "#EC CI_NESTED
      DATA(lv_field) = 'LS_OUTPUT_TMP-TVAL' && ls_monat-low.
      ASSIGN (lv_field) TO <fs_value>.                  "#EC CI_SUBRC
      IF <fs_value> IS ASSIGNED AND <fs_value> IS NOT INITIAL.
        ls_output-tvalx = 0 - <fs_value>.
      ENDIF.

      lv_field = 'LS_OUTPUT_TMP-LVAL' && ls_monat-low.
      ASSIGN (lv_field) TO <fs_value>.                  "#EC CI_SUBRC
      IF <fs_value> IS ASSIGNED AND <fs_value> IS NOT INITIAL.
        ls_output-lvalx = 0 - <fs_value>.
      ENDIF.

      IF ls_output-tvalx IS NOT INITIAL OR ls_output-lvalx IS NOT INITIAL.
        ls_output-rpmax = ls_monat-low.
        APPEND ls_output TO et_output.
        CLEAR:
         ls_output-tvalx,
         ls_output-lvalx.
      ENDIF.

    ENDLOOP.

    IF is_input-monat[] IS INITIAL.
      DO 16 TIMES.                                  "#EC CI_NESTED
        lv_num = lv_num + 1.
        DATA(lv_field2) = 'LS_OUTPUT_TMP-TVAL' && lv_num.
        ASSIGN (lv_field2) TO <fs_value>.               "#EC CI_SUBRC
        IF <fs_value> IS ASSIGNED AND <fs_value> IS NOT INITIAL.
          ls_output-tvalx = 0 - <fs_value>.
        ENDIF.

        lv_field2 = 'LS_OUTPUT_TMP-LVAL' && lv_num.
        ASSIGN (lv_field2) TO <fs_value>.               "#EC CI_SUBRC
        IF <fs_value> IS ASSIGNED AND <fs_value> IS NOT INITIAL.
          ls_output-lvalx = 0 - <fs_value>.
        ENDIF.

        IF ls_output-tvalx IS NOT INITIAL OR ls_output-lvalx IS NOT INITIAL.
          ls_output-rpmax = lv_num.
          APPEND ls_output TO et_output.
          CLEAR:
            ls_output-tvalx,
            ls_output-lvalx.
        ENDIF.
      ENDDO.
      CLEAR:
        lv_num.
```

```
  ENDIF.
 ENDLOOP.

 LOOP AT et_output ASSIGNING FIELD-SYMBOL(<fs_budget>).
  DATA(lv_time) = <fs_budget>-fiscyear && <fs_budget>-rpmax+1(2).
  LOOP AT lt_pi ASSIGNING FIELD-SYMBOL(<fs_pi>) WHERE zdatefrom(6) <= lv_time
                                      AND zdateto(6)   >= lv_time.
   <fs_budget>-pi = <fs_pi>-pi.
   EXIT.
  ENDLOOP.
 ENDLOOP.

ENDFUNCTION.
```

```
*&---------------------------------------------------------------------*
*& Report ZBWR_EXPORT_DATA_TO_CSV
*&---------------------------------------------------------------------*
*&
*&---------------------------------------------------------------------*
REPORT zbwr_export_data_to_csv.

SELECTION-SCREEN  BEGIN OF BLOCK b1 WITH FRAME TITLE TEXT-b01.
 SELECTION-SCREEN SKIP 1.
 PARAMETERS: p_folder    TYPE rlgrap-filename.
 SELECTION-SCREEN SKIP 1.
 SELECTION-SCREEN BEGIN OF LINE.
  PARAMETERS: p_mstr  AS CHECKBOX."CPM Master Data
  SELECTION-SCREEN COMMENT (55) TEXT-001 FOR FIELD p_mstr.
 SELECTION-SCREEN END OF LINE.

 SELECTION-SCREEN BEGIN OF LINE.
  PARAMETERS: p_tmem AS CHECKBOX. "CPM Project Team Member
  SELECTION-SCREEN COMMENT (55) TEXT-002 FOR FIELD p_tmem.
 SELECTION-SCREEN END OF LINE.

 SELECTION-SCREEN BEGIN OF LINE.
  PARAMETERS: p_ccat   AS CHECKBOX."Cost Center Master Data(Attribute Time Dependent)
  SELECTION-SCREEN COMMENT (55) TEXT-003 FOR FIELD p_ccat.
 SELECTION-SCREEN END OF LINE.

 SELECTION-SCREEN BEGIN OF LINE.
  PARAMETERS: p_ccan   AS CHECKBOX."Cost Center Master Data(Attribute Non-Time Dependent)
  SELECTION-SCREEN COMMENT (55) TEXT-004 FOR FIELD p_ccan.
 SELECTION-SCREEN END OF LINE.

 SELECTION-SCREEN BEGIN OF LINE.
  PARAMETERS: p_cctt   AS CHECKBOX."Cost Center Master Data(Text)
  SELECTION-SCREEN COMMENT (55) TEXT-005 FOR FIELD p_cctt.
 SELECTION-SCREEN END OF LINE.

 SELECTION-SCREEN BEGIN OF LINE.
  PARAMETERS: p_cchr   AS CHECKBOX."Cost Center Master Data(Hierarchy)
  SELECTION-SCREEN COMMENT (55) TEXT-006 FOR FIELD p_cchr.
 SELECTION-SCREEN END OF LINE.

 SELECTION-SCREEN BEGIN OF LINE.
  PARAMETERS: p_bpat   AS CHECKBOX."Business Partner Master Data(Attribute Time Dependent)
  SELECTION-SCREEN COMMENT (55) TEXT-007 FOR FIELD p_bpat.
 SELECTION-SCREEN END OF LINE.

 SELECTION-SCREEN BEGIN OF LINE.
  PARAMETERS: p_bpan   AS CHECKBOX."Business Partner Master Data(Attribute Non-Time
Dependent)
  SELECTION-SCREEN COMMENT (55) TEXT-008 FOR FIELD p_bpan.
 SELECTION-SCREEN END OF LINE.
```

```
  SELECTION-SCREEN BEGIN OF LINE.
    PARAMETERS: p_bptt   AS CHECKBOX."Business Partner Master Data(Text)
    SELECTION-SCREEN COMMENT (55) TEXT-010 FOR FIELD p_bptt.
  SELECTION-SCREEN END OF LINE.

  SELECTION-SCREEN BEGIN OF LINE.
    PARAMETERS: p_fptd   AS CHECKBOX."CPM Top Down
    SELECTION-SCREEN COMMENT (55) TEXT-011 FOR FIELD p_fptd.
  SELECTION-SCREEN END OF LINE.

  SELECTION-SCREEN BEGIN OF LINE.
    PARAMETERS: p_frst   AS CHECKBOX."Forecast
    SELECTION-SCREEN COMMENT (55) TEXT-012 FOR FIELD p_frst.
  SELECTION-SCREEN END OF LINE.

  SELECTION-SCREEN BEGIN OF LINE.
    PARAMETERS: p_buex   AS CHECKBOX."Bottom Up - External
    SELECTION-SCREEN COMMENT (55) TEXT-013 FOR FIELD p_buex.
  SELECTION-SCREEN END OF LINE.

  SELECTION-SCREEN BEGIN OF LINE.
    PARAMETERS: p_buin  AS CHECKBOX."Bottom Up - Internal
    SELECTION-SCREEN COMMENT (55) TEXT-014 FOR FIELD p_buin.
  SELECTION-SCREEN END OF LINE.

  SELECTION-SCREEN BEGIN OF LINE.
    PARAMETERS: p_tart  AS CHECKBOX."Target
    SELECTION-SCREEN COMMENT (55) TEXT-015 FOR FIELD p_tart.
  SELECTION-SCREEN END OF LINE.

  SELECTION-SCREEN BEGIN OF LINE.
    PARAMETERS: p_actl   AS CHECKBOX."Actual
    SELECTION-SCREEN COMMENT (55) TEXT-016 FOR FIELD p_actl.
  SELECTION-SCREEN END OF LINE.

SELECTION-SCREEN  END OF BLOCK b1.

AT SELECTION-SCREEN ON VALUE-REQUEST FOR p_folder.

  PERFORM f_select_folder CHANGING p_folder.
*&---------------------------------------------------------------------*
*& Form f_select_folder
*&---------------------------------------------------------------------*
*& text
*&---------------------------------------------------------------------*
*&      <-- P_FOLDER
*&---------------------------------------------------------------------*
FORM f_select_folder  CHANGING cv_folder.
  DATA: lv_string TYPE string,
        lv_title  TYPE string,
        lv_file   TYPE string,
        lv_path   TYPE string.

  lv_title = 'Select Directory'.
  lv_string = 'C:\'.
```

```
  CALL METHOD cl_gui_frontend_services=>directory_browse
    EXPORTING
      window_title        = lv_title
      initial_folder      = lv_string
    CHANGING
      selected_folder     = lv_path
    EXCEPTIONS
      cntl_error          = 1
      error_no_gui        = 2
      not_supported_by_gui = 3
      OTHERS              = 4.

  IF sy-subrc <> 0.
    EXIT.
  ELSE.
    IF lv_path <> space.
      cv_folder = lv_path.
*     CONCATENATE lv_path '\' INTO cv_folder.
    ELSE.
      EXIT.
    ENDIF.
  ENDIF.
ENDFORM.
```

```
 METHOD post_docset_create_entity.
**TRY.
*CALL METHOD SUPER->POST_DOCSET_CREATE_ENTITY
*  EXPORTING
*    IV_ENTITY_NAME        =
*    IV_ENTITY_SET_NAME     =
*    IV_SOURCE_NAME        =
*    IT_KEY_TAB           =
**    io_tech_request_context =
*    IT_NAVIGATION_PATH     =
**    io_data_provider       =
**  IMPORTING
**    er_entity            =
*    .
**  CATCH /iwbep/cx_mgw_busi_exception.
**  CATCH /iwbep/cx_mgw_tech_exception.
**ENDTRY.

   DATA: ls_request_input_data TYPE zcl_zcon_odata_fm_fm_s_mpc=>ts_post_doc,
         lt_message          TYPE bapiret2_t,
         ls_message          TYPE bapiret2,
         lt_msg              TYPE bal_t_msg,
         ls_msg              TYPE bal_s_msg,
         lv_msg              TYPE bapi_msg,
         lv_is_error         TYPE       boolean,
         lt_long_text        TYPE STANDARD TABLE OF bapitgb WITH DEFAULT KEY,
         lv_text             TYPE string,
         lv_message_text      TYPE /iwbep/if_message_container=>ty_s_error_detail,
         lo_message_container  TYPE REF TO     /iwbep/if_message_container,
         lt_doc              TYPE TABLE OF zcont_s_fm_sumrpt.
   io_data_provider->read_entry_data( IMPORTING es_data = ls_request_input_data ).
   /ui2/cl_json=>deserialize( EXPORTING json = ls_request_input_data-is_data CHANGING data =
lt_doc ).
   CALL FUNCTION 'ZFM_DOC_POST'
     EXPORTING
       it_doc     = lt_doc
     IMPORTING
       et_message = lt_message.

**********************************************************************
*  add message to message container
**********************************************************************
   IF NOT lt_message IS INITIAL  AND me->mo_context IS BOUND.
     lo_message_container = me->mo_context->get_message_container( ).

     lv_is_error = abap_false.

     LOOP AT lt_message TRANSPORTING NO FIELDS WHERE type CA 'EAX'.
       lv_is_error = abap_true.
     ENDLOOP.
     IF sy-subrc <> 0.
       READ TABLE lt_message INTO ls_message WITH KEY type = /cpd/cl_sc_cpm_constants=>
gc_success_typ.
       IF sy-subrc = 0.
         er_entity-return_message = ls_message.
```

```
    ENDIF.
  ENDIF.

  LOOP AT lt_message INTO ls_message.
    CALL FUNCTION 'BAPI_MESSAGE_GETDETAIL'
      EXPORTING
        id         = ls_message-id
        number     = ls_message-number
        textformat = 'ASC'
        message_v1 = ls_message-message_v1
        message_v2 = ls_message-message_v2
        message_v3 = ls_message-message_v3
        message_v4 = ls_message-message_v4
        line_size  = 999
      IMPORTING
        message    = lv_msg
      TABLES
        text       = lt_long_text.

    IF lt_long_text IS NOT INITIAL.
      lv_text = lv_msg && cl_abap_char_utilities=>cr_lf.
      LOOP AT lt_long_text ASSIGNING FIELD-SYMBOL(<fs_long_text>).
        lv_text =  lv_text && <fs_long_text>-line && cl_abap_char_utilities=>cr_lf.
      ENDLOOP.
      lv_message_text-message_text = lv_text.
      lv_message_text-severity = convert_type_to_severity( ls_message-type ).
      lo_message_container->add_error_detail(
        EXPORTING
          is_error_detail = lv_message_text
      ).
    ELSE.
      lo_message_container->add_messages_from_bapi(
        EXPORTING
          it_bapi_messages         = VALUE #( ( ls_message ) )
          iv_add_to_response_header = abap_true
      ).
    ENDIF.
  ENDLOOP .

  IF lv_is_error = abap_true.
    RAISE EXCEPTION TYPE /iwbep/cx_mgw_busi_exception
      EXPORTING
        message_container = lo_message_container.
  ENDIF.
 ENDIF.
ENDMETHOD.
```

```
 METHOD post_docset_create_entity.
**TRY.
*CALL METHOD SUPER->POST_DOCSET_CREATE_ENTITY
*  EXPORTING
*    IV_ENTITY_NAME        =
*    IV_ENTITY_SET_NAME    =
*    IV_SOURCE_NAME        =
*    IT_KEY_TAB            =
**    io_tech_request_context =
*    IT_NAVIGATION_PATH    =
**    io_data_provider      =
**  IMPORTING
**    er_entity             =
*    .
**  CATCH /iwbep/cx_mgw_busi_exception.
**  CATCH /iwbep/cx_mgw_tech_exception.
**ENDTRY.

   DATA: ls_request_input_data TYPE zcl_zcon_odata_fm_fm_s_mpc=>ts_post_doc,
         lt_message          TYPE bapiret2_t,
         ls_message          TYPE bapiret2,
         lt_msg              TYPE bal_t_msg,
         ls_msg              TYPE bal_s_msg,
         lv_msg              TYPE bapi_msg,
         lv_is_error         TYPE       boolean,
         lt_long_text        TYPE STANDARD TABLE OF bapitgb WITH DEFAULT KEY,
         lv_text             TYPE string,
         lv_message_text     TYPE /iwbep/if_message_container=>ty_s_error_detail,
         lo_message_container  TYPE REF TO     /iwbep/if_message_container,
         lt_doc              TYPE TABLE OF zcont_s_fm_sumrpt.
   io_data_provider->read_entry_data( IMPORTING es_data = ls_request_input_data ).
   /ui2/cl_json=>deserialize( EXPORTING json = ls_request_input_data-is_data CHANGING data =
lt_doc ).
   CALL FUNCTION 'ZFM_DOC_POST'
     EXPORTING
       it_doc     = lt_doc
     IMPORTING
       et_message = lt_message.
   LOOP AT lt_message INTO ls_message.
     ls_msg-msgty  = ls_message-type.
     ls_msg-MSGid  = ls_message-id.
     ls_msg-msgno  = ls_message-number.
     ls_msg-msgv1  = ls_message-message_v1.
     ls_msg-msgv2  = ls_message-message_v2.
     ls_msg-msgv3  = ls_message-message_v3.
     ls_msg-msgv4  = ls_message-message_v4.
     APPEND ls_msg TO lt_msg.
   ENDLOOP.
*--------------------Raise Exception--------------------------------*
   me->exception_handle(
     EXPORTING
```

```
      it_msg          = lt_msg  " Application Log: Table with Messages
      iv_entity_name   = iv_entity_name
      it_key_tab      = it_key_tab    " table for name value pairs
    IMPORTING
      er_msg_container = lo_message_container
  ).

  READ TABLE lt_msg TRANSPORTING NO FIELDS WITH KEY msgty = /cpd/cl_sc_cpm_constants=>
gc_error_typ.
  IF sy-subrc = 0.
*------ raise business exception if key is not transferred-----------*
    RAISE EXCEPTION TYPE /iwbep/cx_mgw_busi_exception
      EXPORTING
        message_container = lo_message_container
        textid           = /iwbep/cx_mgw_busi_exception=>business_error.
*--------------------------------------------------------------------*
  ENDIF.
  READ TABLE lt_message INTO ls_message WITH KEY type = /cpd/cl_sc_cpm_constants=>
gc_success_typ.
  IF sy-subrc = 0.
    er_entity-return_message = ls_message.
  ENDIF.

 ENDMETHOD.
```

`

CL_INM_NAVIGATION==========CP

INM_APPCC_PPM_OBJECTS

```
method wddomodifyview .

* Constants with data element names, which provide F4 help on selection screen
 constants: cv_dats        type dd04v-datatype value 'DATS'.
 constants: cv_skill_rat    type dd04v-rollname value 'DPR_TV_SKILLS_RATING'.
 constants: cv_skill_id     type dd04v-rollname value 'DPR_TV_SKILLS_ID'.
 constants: cv_date_cont    type string value 'TC_DATE'.
 constants: cv_user_name    type string value '/RPM/TV_USER_NAME'.



 data: lr_label           type ref to  cl_wd_label,
      lr_input_field      type ref to  cl_wd_input_field,
      lr_dropdown_by_idx  type ref to  cl_wd_dropdown_by_idx,
      lr_abstract_inpfield type ref to  cl_wd_abstract_input_field,

      lr_root_view_element type ref to  cl_wd_uielement_container,
      lr_ui_element       type ref to  cl_wd_uielement,  "#EC NEEDED
      ls_ui_ids          like line of wd_this->mt_ui_ids,
      ls_ui_ids_bound     like line of wd_this->mt_ui_ids_bound,
      ls_ui_selected_ddi  like line of wd_this->mt_ui_selected_ddi,
      ls_created_nodes    like line of wd_this->mt_created_nodes,
      lr_display_exit     type ref to  if_wd_context_node,

      lr_node_ddi         type ref to  if_wd_context_node,
      lr_status_node_info  type ref to  if_wd_context_node_info,
      lr_root_node_info    type ref to  if_wd_context_node_info,
      lr_child_node_info   type ref to  if_wd_context_node_info,
      lr_child_1_node_info type ref to  if_wd_context_node_info,
      lr_text_view        type ref to cl_wd_text_view,
      lr_date_range_container type ref to cl_wd_transparent_container,

      lt_child_nodes      type        wdr_context_child_map,
      ls_attribute_info    type        wdr_context_attribute_info.

 data: lv_attribute_path    type string,
      lv_ref_field        type string,
      lv_label_id         type string,
      lv_input_id         type string,
      lv_listbox_id       type string,
      lv_text_id          type string,
      lv_default_value     type string,
      lv_attribute_name    type string,
      lv_refresh          type boole_d,
      lv_root_container_indx type i,
      lv_number           type i,
      lv_child_1_node_name type string.


 data:  ls_shlp_descr       type shlp_descr,
       ls_breadcrumb      type dpr_ts_ui_shlp_bread_crumb,
       ls_fielddescr       type dfies,
       ls_fieldprop        type ddshfprop,
       lt_listbox_values   type dpr_tt_ui_srch_listbox,
       ls_listbox_values   type dpr_ts_ui_srch_listbox,
```

```
      ls_nvp           type dpr_ts_api_name_value_pair,
      lt_nvp           type dpr_tt_api_name_value_pair,
      lv_field_type    type string,
      lv_external_name  type string,
      lv_external_value  type string,
      lv_ddic_type     type dd04v-datatype,
      lv_tmp           type string.

 data:  lr_chglog_node    type ref to if_wd_context_node,
      lr_chglog_element  type ref to if_wd_context_element,
      lv_chglog_name    type string,
      lv_added          type boole_d,              "#EC NEEDED
      lv_length         type i value '20'.

 data:  node_search_in_results  type ref to if_wd_context_node,
      elem_search_in_results  type ref to if_wd_context_element,
      stru_search_in_results  type wd_this->element_search_in_results ,
      item_visible          like stru_search_in_results-visible,
      enable               like stru_search_in_results-enable.
 data lr_date_container type ref to cl_wd_transparent_container.
 data  lv_visibility     type wdy_md_ui_visibility.
 data lo_nd_search_hits type ref to if_wd_context_node.
 data lo_el_search_hits type ref to if_wd_context_element.

 constants lc_dashboard type /rpm/tv_search_type value '04'.
 constants lc_interface type /rpm/tv_search_type value '02'.

 data lr_btn type ref to cl_wd_button. " note_1634727 1/2
 data lv_object_type  type ddshdefval.                  "note 1704870 1/3

* In case of list box adaption: only partly context changes necessary

 if wd_comp_controller->mv_search_type eq lc_dashboard.
  build_confirmation_buttons( iv_view = view iv_search_type = wd_comp_controller->
mv_search_type ).

*   Set the hits default value to 400
  lo_nd_search_hits = wd_context->get_child_node( name = wd_this->wdctx_search_hits ).
  lo_el_search_hits = lo_nd_search_hits->get_element( ).
*    set single attribute
  lo_el_search_hits->set_attribute(
    name =  `HITS`
    value = '400' ).

 endif.


 if wd_this->mc_listbox_adaption = abap_true.
  wd_this->listbox_adaption( ).

  call method wd_this->mr_component_ctrl->mr_ui_log_search->get_listbox_values_all
    importing
      et_helpvalues_all = lt_listbox_values.

  wd_this->mt_listbox_values = lt_listbox_values.
  if not  wd_this->mr_node_status is initial.
    call method wd_this->mr_node_status->set_lead_selection_index
```

```
      exporting
        index = 1.

    lr_status_node_info =  wd_this->mr_node_status->get_node_info( ).
    lv_chglog_name  = lr_status_node_info->get_name( ) .
    lr_chglog_node  = wd_context->get_child_node( name = lv_chglog_name  ).
    lr_chglog_element = lr_chglog_node->get_element( ) .


    call method lr_chglog_element->get_attribute
      exporting
        name  = 'NAME'
      importing
        value = lv_attribute_name.

    read table wd_this->mt_ui_selected_ddi into ls_ui_selected_ddi
    with key  name = 'STATUS'.
    if sy-subrc = 0.
      delete table wd_this->mt_ui_selected_ddi from ls_ui_selected_ddi.
      clear ls_ui_selected_ddi.
    endif.
* Save: The actual selected DDI is aways added into line 1
    ls_ui_selected_ddi-name  = 'STATUS'.
    ls_ui_selected_ddi-value = lv_attribute_name.
    insert ls_ui_selected_ddi into wd_this->mt_ui_selected_ddi index 1.
   endif.
   exit.
  endif.

* Set Search in Result button visibility.
  node_search_in_results = wd_context->get_child_node( name = wd_this->
wdctx_search_in_results ).
  elem_search_in_results = node_search_in_results->get_element(  ).



*/
*/ note_1634727 2/2 begin
if wd_comp_controller->mv_turn_off_srch_in_res = abap_true.
   wd_this->mr_component_ctrl->mr_ui_log_search->clear_breadcrumb( ).
   wd_comp_controller->mv_turn_off_srch_in_res = abap_false.
  endif.
*/ note_1634727 2/2 end
*/

  item_visible = wd_this->mr_component_ctrl->mr_ui_log_search->get_search_in_result_visible( ).

  elem_search_in_results->set_attribute(
   exporting
    name =  `VISIBLE`
    value = item_visible ).

  if wd_comp_controller->mt_search_result is not initial.
   elem_search_in_results->set_attribute(
    exporting
     name =  `ENABLE`
     value = abap_true ).
```

```
 else.
  elem_search_in_results->set_attribute(
   exporting
    name =  `ENABLE`
    value = abap_false ).
 endif.

* In case of no changes -> WDDOMODIFYVIEW has to be interrupted
 if wd_this->mr_component_ctrl->mv_modify_srch_crt = abap_false.
  exit.
 endif.

* Initialize
 clear: wd_this->mr_component_ctrl->ms_shlp,
       wd_this->mt_listbox_values.
* Initialize selected values of dropdown lists
 clear wd_this->mt_ui_selected_ddi.

 lv_root_container_indx = 1.  " number of static context elements

* Trigger PRESEL step
* for external search call
 case wd_this->mr_component_ctrl->mv_called_external.
  when abap_true.
   case wd_this->mr_component_ctrl->mv_object_type.
    when cl_dpr_co=>sc_ot_object_link.
     call method wd_this->mr_component_ctrl->mr_ui_log_search->service_external_call
      exporting
       iv_object_type     = cl_dpr_co=>sc_ot_object_link
       iv_version_type    = wd_this->mr_component_ctrl->mv_version_type
       it_screen_interface = wd_this->mr_component_ctrl->mt_screen_interface
       it_identifier      = wd_this->mr_component_ctrl->mt_identifier
      importing
       ev_shelp_name      = wd_this->mr_component_ctrl->mv_select_simple_itm.
   endcase.

* for search (internal call)
  when abap_false.
   call method wd_this->mr_component_ctrl->mr_ui_log_search->controll_search
    exporting
     iv_shelp_name = wd_this->mr_component_ctrl->mv_select_simple_itm
     iv_step_name  = cl_dpr_ui_log_search=>sc_trigger_presel.

 endcase.

* Get search help infos from logic class
 if wd_comp_controller->mv_breadcrumb_sel = abap_false.
  call method wd_this->mr_component_ctrl->mr_ui_log_search->get_shlp_info
   receiving
    rs_shlp_descr = ls_shlp_descr.
 else.
  call method wd_this->mr_component_ctrl->mr_ui_log_search->get_current_breadcrumb
   importing
    es_breadcrumb = ls_breadcrumb.
  ls_shlp_descr = ls_breadcrumb-shlp_desc.
  wd_this->modify_shlp_descr(  exporting iv_view = view
                changing cv_shlp_descr = ls_shlp_descr   ).
```

```
 endif.


 call method wd_this->mr_component_ctrl->mr_ui_log_search->get_listbox_values_all
  importing
    et_helpvalues_all = lt_listbox_values.

 wd_this->mt_listbox_values = lt_listbox_values.

* Save search help info
 wd_this->mr_component_ctrl->ms_shlp = ls_shlp_descr.
 sort ls_shlp_descr-fieldprop by shlpselpos.

* Get parent UI element (Transparent Container)
 lr_root_view_element ?= view->get_root_element( ) .
 lr_root_node_info     = wd_context->get_node_info( ) .


************************************************************
 if first_time = abap_false.                    "FALSE
************************************************************
   call method wd_context->get_child_nodes
    receiving
     child_nodes = lt_child_nodes.

* This number corresponds to the number of STATIC context nodes
   describe table lt_child_nodes lines lv_number.
   if lv_number > 1.

* Delete node info
    loop at wd_this->mt_created_nodes into ls_created_nodes.

      call method lr_root_node_info->remove_child_node
       exporting
         name = ls_created_nodes.
    endloop.

    clear: wd_this->mt_created_nodes.

* Delete UI elements
    loop at wd_this->mt_ui_ids into ls_ui_ids.
      call method lr_root_view_element->remove_child
       exporting
         id       = ls_ui_ids
       receiving
         the_child = lr_ui_element.
    endloop.

    lv_refresh = abap_true.
    clear: wd_this->mt_ui_ids, wd_this->mt_ui_ids_bound.

  endif.
 endif.

************************************************************
 if first_time = abap_true                      "TRUE
 or lv_refresh = abap_true.
```

```
************************************************************
* Create dynamic context node SEARCH_CRITERIA
   call method lr_root_node_info->add_new_child_node
    exporting
     name                    = wd_this->mc_search_criteria
     is_mandatory            = abap_true
     is_mandatory_selection  = abap_false
     is_multiple             = abap_false
     is_multiple_selection   = abap_false
     is_singleton            = abap_true
     is_initialize_lead_selection = abap_true
     is_static               = abap_false
    receiving
     child_node_info         = lr_child_node_info.

   ls_created_nodes = wd_this->mc_search_criteria.
   append ls_created_nodes to wd_this->mt_created_nodes.

   wd_this->modify_shlp_descr(  exporting iv_view = view
                      changing cv_shlp_descr = ls_shlp_descr   ).


****************************************************
*/  Create node attributes, layouts & UI elements
****************************************************
* Start of note 1704870 2/3
   read table ls_shlp_descr-fieldprop with key fieldname = 'OBJECT_TYPE' into ls_fieldprop.
   if sy-subrc is initial.
    lv_object_type = ls_fieldprop-defaultval.
   endif.
* End of note 1704870 2/3

   loop at ls_shlp_descr-fieldprop into ls_fieldprop.
    if ls_fieldprop-shlpselpos is initial.
     continue.
    endif.

    lv_field_type = wd_this->mc_input_field.   " input field is default

    read table ls_shlp_descr-fielddescr
    into ls_fielddescr with key fieldname = ls_fieldprop-fieldname.

    wd_this->get_ddic_type( exporting  iv_name =    ls_fielddescr-rollname
                importing  ev_ddic_type =    lv_ddic_type ).

    if lv_ddic_type eq cv_dats.
*       Create date container only the first time
      if lr_date_container is not bound.
       lr_date_container = cl_wd_transparent_container=>new_transparent_container( id  =
cv_date_cont
                                                    view  = view ).
       append cv_date_cont to wd_this->mt_ui_ids.
       cl_wd_matrix_layout=>new_matrix_layout( container = lr_date_container ).
       cl_wd_matrix_head_data=>new_matrix_head_data( col_span = 4 element  =
lr_date_container ).
       lv_root_container_indx = lv_root_container_indx + 1.
       lr_root_view_element->add_child( index     = lv_root_container_indx
```

```
                                the_child = lr_date_container ).
      endif.

    endif.


*>>>> LABEL
    lv_root_container_indx = lv_root_container_indx + 1.

    concatenate 'INP' ls_fieldprop-fieldname
    into lv_input_id separated by '_'.
    translate lv_input_id to upper case .

    lv_external_name = ls_fieldprop-fieldname.
*/
*/ note_1633977 begin
    if lv_external_name = 'PROJECT_DESCRIPT'.
      lv_default_value = cl_bsp_get_text_by_alias=>get_text( alias =
'DEVELOPMENT_PROJECTS_UI_WD/PROJECT_NAME' language = sy-langu ).
* Start of note 1704870 3/3
    elseif lv_external_name = 'EXTERNAL_ID' and
         lv_object_type cs 'DPT'.
      lv_default_value = cl_bsp_get_text_by_alias=>get_text( alias =
'DEVELOPMENT_PROJECTS_UI_WD/PROJECT_TEMPLATE_NUMBER' language = sy-langu ).
    elseif lv_external_name = 'TEXTU' and
         lv_object_type cs 'DPT'.
      lv_default_value = cl_bsp_get_text_by_alias=>get_text( alias =
'DEVELOPMENT_PROJECTS_UI_WD/PROJECT_TEMPLATE_NAME' language = sy-langu ).
* End of note 1704870 3/3
    else.
      lv_default_value = ls_fielddescr-scrtext_l.
    endif.
*/ note_1633977 end
*/
    if not ( lv_ddic_type eq cv_dats and ( ls_fieldprop-fieldname cs '_H' or ls_fieldprop-fieldname cs
'FINISH' ) ).

      concatenate 'LBL' ls_fieldprop-fieldname
      into lv_label_id separated by '_'.
      translate lv_label_id to upper case .

* Create UI element: label
      call method cl_wd_label=>new_label
        exporting
          id       = lv_label_id
          label_for = lv_input_id
          text     = lv_default_value
          view     = view
*         width    = '18%'
*         wrapping  = abap_true
        receiving
          control   = lr_label.

*   Create layout
      call method cl_wd_matrix_head_data=>new_matrix_head_data
        exporting
          col_span = 1
```

```
          element  = lr_label
          width    = '18%'.

*       lr_label->set_wrapping( 'X' ).

* Add label to parent view container
       if lv_ddic_type ne cv_dats.

         call method lr_root_view_element->add_child
           exporting
             index     = lv_root_container_indx
             the_child = lr_label.
       else.

         call method lr_date_container->add_child
           exporting
*            index     = lv_root_container_indx
             the_child = lr_label.
         lv_root_container_indx = lv_root_container_indx - 1 .
       endif.

*       Create a separate container for date ranges
       if lv_ddic_type eq cv_dats.

         data lv_id type string.
         lr_date_range_container = cl_wd_transparent_container=>new_transparent_container( view
= view ).
         lv_id = lr_date_range_container->if_wd_view_element~get_id( ).
         append lv_id  to wd_this->mt_ui_ids.
         cl_wd_flow_layout=>new_flow_layout(  container = lr_date_range_container  ).
         cl_wd_matrix_data=>new_matrix_data( col_span = '3' element  = lr_date_range_container
width = '82%' ).
         lr_date_container->add_child( the_child = lr_date_range_container ).
       endif.

     else.
*       Coding for melting groups for dates

       if ls_fieldprop-fieldname cs '_H' .
         concatenate  'SEP' ls_fieldprop-fieldname
         into lv_text_id separated by '_'.
         lr_text_view =  cl_wd_text_view=>new_text_view( id = lv_text_id
                                         text = '-'
                                         view = view  )   .
         append lv_text_id to wd_this->mt_ui_ids.

       elseif ls_fieldprop-fieldname cs 'FINISH'.
         concatenate  'SLH' ls_fieldprop-fieldname
         into lv_text_id separated by '_'.
         lr_text_view =  cl_wd_text_view=>new_text_view( id = lv_text_id
                                         text = '/'
                                         view = view  )   .
         append lv_text_id to wd_this->mt_ui_ids.
       endif.

       cl_wd_flow_data=>new_flow_data( cell_design = '00' "cl_wd_flow_data=>e_cell_design-rpad
                             element  = lr_text_view ).
```

```
      lr_date_range_container->add_child( lr_text_view ).
      lv_root_container_indx = lv_root_container_indx - 1 .

    endif.
*>>>> INPUT FIELD or DROPDOWN LISTBOX

    lv_root_container_indx = lv_root_container_indx + 1.

* Check field type (input field or dropdown listbox?)
    if not lt_listbox_values is initial.
      read table lt_listbox_values into ls_listbox_values with key
      field_name = ls_fieldprop-fieldname.

      if sy-subrc = 0.          " dropdown list
        clear lt_nvp.
        lv_field_type = wd_this->mc_listbox.
        lt_nvp       = ls_listbox_values-nv_pair.
      endif.
    endif.


    case lv_field_type.
*********************************
      when  wd_this->mc_listbox.
*********************************
*         lv_sy_tabix = sy-tabix.
        concatenate wd_this->mc_help_values lv_input_id+4
        into lv_child_1_node_name separated by '_'.
        lv_listbox_id = lv_child_1_node_name.

* Create dynamic context node for listboxes (HELP_VALUES)
        call method lr_root_node_info->add_new_child_node
          exporting
            name                    = lv_child_1_node_name
            is_mandatory              = abap_true
            is_mandatory_selection      = abap_false
            is_multiple               = abap_true
            is_multiple_selection       = abap_false
            is_singleton              = abap_true
            is_initialize_lead_selection = abap_true
            is_static                 = abap_false
          receiving
            child_node_info           = lr_child_1_node_info.

* Help table needed for deletion
        ls_created_nodes = lv_child_1_node_name.
        append ls_created_nodes to wd_this->mt_created_nodes.


* Fill context node attributes info (ls_attribute_info)
        ls_attribute_info-name      = wd_this->mc_nvp_name.
        ls_attribute_info-type_name  = 'STRING'.
        ls_attribute_info-node_info  = lr_child_1_node_info.
        ls_attribute_info-is_static  = abap_false.
        ls_attribute_info-value_help_mode = 0.
```

```
* Add attribute (NAME) to context node
      call method lr_child_1_node_info->add_attribute
        exporting
          attribute_info = ls_attribute_info.

* Fill context node attributes info (ls_attribute_info)
      ls_attribute_info-name      = wd_this->mc_nvp_value.
      ls_attribute_info-type_name  = 'STRING'.
      ls_attribute_info-node_info  = lr_child_1_node_info.
      ls_attribute_info-is_static = abap_false.
      ls_attribute_info-value_help_mode = 1.

* Add attribute (VALUE) to context node
      call method lr_child_1_node_info->add_attribute
        exporting
          attribute_info = ls_attribute_info.

      concatenate lv_child_1_node_name wd_this->mc_nvp_value
      into lv_attribute_path separated by '.'.

* Create UI element: dropdown list
      call method cl_wd_dropdown_by_idx=>new_dropdown_by_idx
        exporting
          bind_texts = lv_attribute_path
          view       = view
          id         = lv_listbox_id
          on_select  = 'ONSELECT_DDI'
        receiving
          control    = lr_dropdown_by_idx.

* Create layout data: dropdown list
      call method cl_wd_matrix_data=>new_matrix_data
        exporting
          col_span = 3
          element  = lr_dropdown_by_idx
          width    = '82%'.

* Add to parent view container
      call method lr_root_view_element->add_child
        exporting
          index     = lv_root_container_indx
          the_child = lr_dropdown_by_idx.

* Label must be updated
      call method lr_label->set_label_for
        exporting
          value = lv_listbox_id.

      concatenate wd_this->mc_search_criteria
                  lv_child_1_node_name
                  into lv_attribute_path separated by '.'.

      lr_node_ddi  = wd_context->get_child_node( name = lv_child_1_node_name  ).

* Bind table
      call method lr_node_ddi->bind_table
        exporting
```

```
          new_items = lt_nvp.


*        It's important, to save only one line for each dropdown list
        if not lt_nvp[] is initial.
         read table lt_nvp into ls_nvp index 1.

         read table wd_this->mt_ui_selected_ddi into ls_ui_selected_ddi
         with key  name = ls_nvp-name.
         if sy-subrc = 0.
          delete table wd_this->mt_ui_selected_ddi from ls_ui_selected_ddi.
          clear ls_ui_selected_ddi.
         endif.
*        Save: The actual selected DDI is aways added into line 1
         ls_ui_selected_ddi-name  = ls_fieldprop-fieldname.
         ls_ui_selected_ddi-value = ls_nvp-name.
         insert ls_ui_selected_ddi into table wd_this->mt_ui_selected_ddi.
        endif.

* DDL with preselected value: SET LEAD SELECTION
        if lv_child_1_node_name cs 'STATUS' and lv_child_1_node_name ns 'SUP' and lt_nvp is not
initial.
         wd_this->mr_node_status = lr_node_ddi.
        endif.


* Save a list of bound attributes for further use
        append lv_label_id   to wd_this->mt_ui_ids.
        append lv_listbox_id to wd_this->mt_ui_ids.

        ls_ui_ids_bound-id =   lv_listbox_id.
        ls_ui_ids_bound-field_name = ls_fieldprop-fieldname.
        append ls_ui_ids_bound to wd_this->mt_ui_ids_bound.

        clear: ls_ui_ids_bound, lv_label_id, lv_listbox_id, lv_input_id.

**********************************
      when wd_this->mc_input_field.
**********************************
* Fill context node attributes info (ls_attribute_info)
        ls_attribute_info-name  = lv_input_id.
        ls_attribute_info-type_name  = ls_fielddescr-rollname.
        ls_attribute_info-node_info  = lr_child_node_info.
        ls_attribute_info-is_static = abap_false.

* Check for external input data
        if wd_this->mr_component_ctrl->mv_called_external = abap_true.

         wd_this->transfer_external_input(
          exporting
            name =   lv_external_name        " String
          importing
            value =   lv_external_value  " String
         ).


         ls_attribute_info-default_value = lv_external_value.
```

```
        endif.
        concatenate ls_fielddescr-tabname ls_fielddescr-fieldname
        into lv_ref_field separated by '-'.
        ls_attribute_info-reference_field = lv_ref_field.

* Provide F4 help for some special fields
        wd_this->get_ddic_type(
          exporting
            iv_name =    ls_fielddescr-rollname
         importing
            ev_ddic_type =    lv_ddic_type ).

        if lv_ddic_type = cv_dats or ls_fielddescr-rollname = cv_user_name.
          ls_attribute_info-value_help_mode = 0.
          lv_length = 8.
        else.
          ls_attribute_info-value_help_mode = 1.
        endif.

        if ls_fielddescr-rollname = cv_skill_rat or
           ls_fielddescr-rollname = cv_skill_id.
          ls_attribute_info-value_help_mode = 0.
        endif.

* Add attribute to context node.
        call method lr_child_node_info->add_attribute
          exporting
            attribute_info = ls_attribute_info.

* Store external input within change log
        lv_chglog_name  = lr_child_node_info->get_name( ) .
        lr_chglog_node  = wd_context->get_child_node( name = lv_chglog_name  ).
        if not lv_external_value is initial.
          lr_chglog_element = lr_chglog_node->get_element( ) .

          call method wd_this->mr_change_log->add_context_attribute_change
            exporting
              element       = lr_chglog_element
              attribute_name = ls_attribute_info-name
              new_value     = ls_attribute_info-default_value
              force_entry    = abap_true
            receiving
              entry_added    = lv_added.

        endif.

        concatenate wd_this->mc_search_criteria lv_input_id into lv_attribute_path
        separated by '.'.

* Create UI element: input field
        call method cl_wd_input_field=>new_input_field
          exporting
            bind_value = lv_attribute_path
            id        = lv_input_id
            view      = view
            length     = lv_length
          receiving
```

```
          control    = lr_input_field.
        lv_length = 20.
* Create layout data: input field
        if lr_date_range_container is bound and lv_ddic_type eq cv_dats.
         call method cl_wd_flow_data=>new_flow_data
          exporting
           cell_design = '00' "cl_wd_flow_data=>e_design-rpad
           element     = lr_input_field.
*           width    = '82%'.
        else.
         call method cl_wd_matrix_data=>new_matrix_data
          exporting
           col_span = 3
           element  = lr_input_field
           width    = '82%'.
        endif.

* Create event OnEnter
        lr_abstract_inpfield ?= lr_input_field.
        call method lr_abstract_inpfield->set_on_enter
         exporting
          value = 'ONBUTTON_SEARCH'.


* Add to parent view container

        if lv_ddic_type ne cv_dats.
         call method lr_root_view_element->add_child
          exporting
           index     = lv_root_container_indx
           the_child = lr_input_field.
        else.
         call method lr_date_range_container->add_child
          exporting
*            index     = lv_root_container_indx
           the_child = lr_input_field.
         lv_root_container_indx = lv_root_container_indx - 1 .
*          lr_date_range_container_old = lr_date_range_container.
        endif.

* Save a list of bound attributes for further use
        append lv_label_id   to wd_this->mt_ui_ids.
        append lv_input_id   to wd_this->mt_ui_ids.
        append lv_listbox_id to wd_this->mt_ui_ids.

        ls_ui_ids_bound-id =   lv_input_id.
        ls_ui_ids_bound-field_name = ls_fieldprop-fieldname.
        append ls_ui_ids_bound to wd_this->mt_ui_ids_bound.
        clear: ls_ui_ids_bound, lv_label_id, lv_listbox_id, lv_input_id.

     endcase.

   endloop.

* Always add the static attribute 'HITS'
   ls_ui_ids_bound-id =         wd_this->mc_hits.
   ls_ui_ids_bound-field_name =  wd_this->mc_hits.
```

```
   append ls_ui_ids_bound to wd_this->mt_ui_ids_bound.

 endif.

****************************
*      Buttons
****************************
****/
****/ note_1634727 2/2 begin
***  if wd_comp_controller->mv_turn_off_srch_in_res = abap_true.
***    lr_btn ?=  view->get_element( id = 'BTN_SEARCH_IN_RESULTS' ).
***    lr_btn->set_visible( '01' ).
***    wd_this->mr_component_ctrl->mv_turn_off_srch_in_res = abap_false.
***  endif.
****/ note_1634727 2/2 end
****/

* 'Exit'
 lr_display_exit = wd_context->get_child_node( name = `DISPLAY_EXIT` ).
 call method lr_display_exit->get_attribute
   exporting
     name  = 'DISPLAY'
   importing
     value = lv_visibility.

 case wd_this->mr_component_ctrl->mv_display_exit.
   when abap_false.
     if lv_visibility <> cl_wd_uielement=>e_visible-none.
       call method lr_display_exit->set_attribute
         exporting
           value = cl_wd_uielement=>e_visible-none
           name  = 'DISPLAY'.
     endif.
   when abap_true.
     if lv_visibility <> cl_wd_uielement=>e_visible-visible.
       call method lr_display_exit->set_attribute
         exporting
           value = cl_wd_uielement=>e_visible-visible
           name  = 'DISPLAY'.
     endif.
 endcase.


* Reset mv_modify_view
 wd_this->mr_component_ctrl->mv_modify_srch_crt = abap_false.

endmethod.
```