\*&---------------------------------------------------------------------\*  
\*& Report  YPROCOWM  
\*&  
\*&---------------------------------------------------------------------\*  
\*&  
\*&  
\*&---------------------------------------------------------------------\*  
  
REPORT YPROCOWM NO STANDARD PAGE HEADING  
                                   LINE-SIZE 170  
                                   LINE-COUNT 58  
                                   MESSAGE-ID ZPSDVRP.  
TABLES: PROJ,PRPS.  
\*-----------------------------------------------------------------------  
\* T Y P E S  
\*-----------------------------------------------------------------------  
TYPES:  
  BEGIN OF \_PROJ,  
    VBUKR TYPE PS\_VBUKR,  
    VKOKR TYPE PS\_VKOKR,  
    PSPNR TYPE PS\_INTNR,  
    PSPID TYPE PS\_PSPID,  
    POST1 TYPE PS\_POST1,  
    PLSEZ TYPE PS\_PLSEZ,  
    OBJNR TYPE J\_OBJNR,  
  END OF \_PROJ,  
  
  BEGIN OF \_PRPS,  
    PSPNR TYPE PS\_POSNR,  
    POSID TYPE PS\_POSID,  
    POSKI TYPE PS\_POSKI,  
    PRCTR TYPE PRCTR,  
    PRART TYPE PS\_PRART,  
    POST1 TYPE PS\_POST1,  
    PSPHI TYPE PS\_PSPHI,  
    OBJNR TYPE J\_OBJNR,  
    USR00 TYPE USR00PRPS,  
    ERDAT TYPE ERDAT,  
    STUFE TYPE PS\_STUFE,  
  END OF \_PRPS,  
  
  BEGIN OF \_RPSCO,  
    OBJNR TYPE J\_OBJNR,  
    WRTTP TYPE CO\_WRTTP,  
    GJAHR TYPE GJAHR,  
    VORGA TYPE BP\_VORGANG,  
    VERSN TYPE BP\_VERSION,  
    BELTP TYPE BP\_INOUT,  
    WLP00 TYPE BP\_WPL,  
  END OF \_RPSCO,  
  
  BEGIN OF \_OUTPUT,  
    COMP\_NAME  TYPE NAME\_1,  
    STEP\_NAME(30),  
    PROJ\_PSPID TYPE PS\_PSPID,  
    PROJ\_POST1 TYPE PS\_POST1,  
    PROJ\_PLSEZ TYPE PS\_PLSEZ,  
    PRPS\_USR00 TYPE USR00PRPS,  
    POSID      TYPE PS\_POSID,  
    PRPS\_POST1 TYPE PS\_POST1,  
    PROJ\_TYPE  TYPE PS\_PRATX,  
    STATUS     TYPE J\_INACT,  
    TASK\_STAT  TYPE J\_STEXT,  
    PROJ\_STAT  TYPE J\_STEXT,  
    TASK\_DATE  TYPE ERDAT,  
    VBUKR TYPE PS\_VBUKR,  
    VKOKR TYPE PS\_VKOKR,  
    PSPNR TYPE PS\_INTNR,  
    PRPS\_POSKI TYPE PS\_POSKI,  
    PRCTR TYPE PRCTR,  
    PSPHI TYPE PS\_PSPHI,  
    OBJNR TYPE J\_OBJNR,  
    KUNNR TYPE NAME1\_GP,  
    PARVW TYPE VTXTK,  
    SLS\_PRICE  TYPE BP\_WPL,  
    ORIG\_COST  TYPE BP\_WPL,  
    CURR\_BUDG  TYPE BP\_WPL,  
    END\_COST   TYPE BP\_WPL,  
    ORDR\_COST  TYPE BP\_WPL,  
    ACTL\_COST  TYPE BP\_WPL,  
    ORDR\_PERC(20)  TYPE C,  
    CURR\_ACTL  TYPE BP\_WPL,  
    DEPT(1),  
    LEVEL TYPE I,  
  END OF \_OUTPUT.  
CONSTANTS:C\_MESS\_CL   TYPE SY-MSGID        VALUE 'ZPSDVRP',  
          C\_E         TYPE BAPIRETURN-TYPE VALUE 'E',  
          C\_A         TYPE BAPIRETURN-TYPE VALUE 'A',  
          C\_F         TYPE C               VALUE 'F',  
          C\_I         TYPE BAPIRETURN-TYPE VALUE 'I',  
          C\_X         TYPE C               VALUE 'X',  
          C\_W         TYPE BAPIRETURN-TYPE VALUE 'W',  
          C\_CSV(5)                         VALUE '\*.CSV',  
          C\_SEP       TYPE C               VALUE ',',  
          C\_PERIOD    TYPE C               VALUE '.',  
          C\_COMMA     TYPE C               VALUE ',',  
          C\_1         TYPE C               VALUE '1',  
          C\_%(1)      TYPE C               VALUE '%',  
          C\_01(2)     TYPE C               VALUE '01',  
          C\_21(2)     TYPE C               VALUE '21',  
          C\_02(2)     TYPE C               VALUE '02',  
          C\_04(2)     TYPE C               VALUE '04',  
          C\_22(2)     TYPE C               VALUE '22',  
          C\_42(2)     TYPE C               VALUE '42',  
          C\_2         TYPE C               VALUE '2',  
          C\_001(3)                         VALUE '001',  
          C\_002(3)                         VALUE '002',  
          C\_I0067(5)                       VALUE 'I0067',  
          C\_I0043(5)                       VALUE 'I0043',  
          C\_I0045(5)                       VALUE 'I0045',  
          C\_I0046(5)                       VALUE 'I0046',  
          C\_ZERO(3)   TYPE C               VALUE '000',  
          C\_0000(4)   TYPE C               VALUE '0000',  
          C\_NUMC(4)                        VALUE 'NUMC',  
          C\_KSTR(4)   TYPE C               VALUE 'KSTR',  
          C\_KSTP(4)   TYPE C               VALUE 'KSTP'.  
\*----------------------------------------------------------------------\*  
\* SELECTION-SCREEN  
\*----------------------------------------------------------------------\*  
  
SELECTION-SCREEN BEGIN OF BLOCK SELECTION  
                        WITH FRAME TITLE TEXT-001.  
PARAMETERS:  
  P\_VBUKR   TYPE PROJ-VBUKR OBLIGATORY DEFAULT '1000',  
  P\_VKOKR   TYPE PROJ-VKOKR OBLIGATORY DEFAULT '1000',  
  P\_STAT    TYPE JEST-STAT.  
  
SELECT-OPTIONS:  
  S\_PSPID   FOR PROJ-PSPID DEFAULT 'A.OH00000',  
  S\_POSID   FOR PRPS-POSID,  
  S\_POSKI   FOR PRPS-POSKI,  
  S\_PRCTR   FOR PRPS-PRCTR,  
  S\_PRART   FOR PRPS-PRART.  
SELECTION-SCREEN END OF BLOCK SELECTION.  
  
SELECTION-SCREEN BEGIN OF BLOCK DOWNLOAD  
                          WITH FRAME TITLE TEXT-002.  
PARAMETERS:  
  P\_EXPATH TYPE RLGRAP-FILENAME.  
SELECTION-SCREEN END OF BLOCK DOWNLOAD.  
  
INCLUDE  ZCLASS\_LOG\_UTILITY.  
CLASS LCL\_EVENTS DEFINITION.  
  PUBLIC SECTION.  
    METHODS:  
      ON\_DOUBLE\_CLICK FOR EVENT DOUBLE\_CLICK OF CL\_SALV\_EVENTS\_TREE  
        IMPORTING NODE\_KEY COLUMNNAME ,  
      ON\_LINK\_CLICK FOR EVENT LINK\_CLICK OF CL\_SALV\_EVENTS\_TREE  
        IMPORTING COLUMNNAME,  
      ON\_BEFORE\_USER\_COMMAND FOR EVENT BEFORE\_SALV\_FUNCTION OF CL\_SALV\_EVENTS  
        IMPORTING E\_SALV\_FUNCTION,  
      ON\_AFTER\_USER\_COMMAND FOR EVENT AFTER\_SALV\_FUNCTION OF CL\_SALV\_EVENTS  
        IMPORTING E\_SALV\_FUNCTION,  
      ON\_KEYPRESS FOR EVENT KEYPRESS OF CL\_SALV\_EVENTS\_TREE  
        IMPORTING NODE\_KEY COLUMNNAME KEY.  
    DATA: T\_OTREE TYPE TABLE OF \_OUTPUT.  
ENDCLASS.  
CLASS LCL\_EVENTS IMPLEMENTATION.  
  METHOD ON\_DOUBLE\_CLICK.  
    DATA LS\_OTREE TYPE \_OUTPUT.  
    READ TABLE T\_OTREE INTO LS\_OTREE INDEX NODE\_KEY.  
    IF SY-SUBRC = 0.  
      CASE COLUMNNAME.  
        WHEN 'PROJ\_PSPID'.  
          SET PARAMETER ID 'PSP' FIELD LS\_OTREE-PROJ\_PSPID.  
          CALL TRANSACTION 'CJ2A' AND SKIP FIRST SCREEN.  
      ENDCASE.  
    ENDIF.  
  ENDMETHOD.  
  
  METHOD ON\_LINK\_CLICK.  
  ENDMETHOD.                    "  
  
  METHOD ON\_BEFORE\_USER\_COMMAND.  
  ENDMETHOD.  
  
  METHOD ON\_AFTER\_USER\_COMMAND.  
  ENDMETHOD.  
  
  METHOD ON\_KEYPRESS.  
  ENDMETHOD.  
  
ENDCLASS.  
CLASS LCL\_VIEW DEFINITION.  
  PUBLIC SECTION.  
    CLASS-DATA R\_VIEW TYPE REF TO LCL\_VIEW.  
    METHODS:  
      CONSTRUCTOR,  
      DISP\_TREE IMPORTING T\_OP TYPE STANDARD TABLE  
                          COMP\_NAME TYPE NAME\_1,  
      F4\_HLPFILE CHANGING VALUE(EXPATH) TYPE RLGRAP-FILENAME,  
      GET\_PARAMS EXPORTING  
                  VALUE(VBUKR) TYPE PROJ-VBUKR  
                  VALUE(VKOKR) TYPE PROJ-VKOKR  
                  VALUE(STAT)  TYPE JEST-STAT  
                  VALUE(EXPATH) TYPE RLGRAP-FILENAME  
                  VALUE(PSPID) TYPE RSELOPTION  
                  VALUE(POSID) TYPE RSELOPTION  
                  VALUE(POSKI) TYPE RSELOPTION  
                  VALUE(PRCTR) TYPE RSELOPTION  
                  VALUE(PRART) TYPE RSELOPTION ,  
      SAVE\_FILE EXPORTING VALUE(T\_LOG) TYPE BAPIRET2\_T ,  
      CHECK\_FILE EXPORTING VALUE(T\_LOG) TYPE BAPIRET2\_T,  
      WRITE\_LOG IMPORTING VALUE(EXPATH) TYPE RLGRAP-FILENAME  
                EXPORTING VALUE(T\_LOG) TYPE BAPIRET2\_T .  
  
    DATA:  
      V\_COMP\_NAME TYPE NAME\_1,  
      R\_TREE   TYPE REF TO CL\_SALV\_TREE,  
      R\_EVENTS TYPE REF TO LCL\_EVENTS,  
      T\_OUTPUT TYPE TABLE OF \_OUTPUT,  
      T\_OTREE TYPE TABLE OF \_OUTPUT,  
      I\_RESULT TYPE TABLE OF SOLISTI1.  
  PRIVATE SECTION.  
    METHODS:  
      ADD\_LEVEL IMPORTING LI\_LINE TYPE \_OUTPUT  
                          LI\_KEY TYPE LVC\_NKEY  
               EXPORTING LO\_KEY TYPE LVC\_NKEY,  
      BUILD\_TREE,  
      CREATE\_CSV\_HDR,  
      CREATE\_TREE,  
      CREATE\_METAINF,  
      EDIT\_COLUMNS,  
      GET\_FUNCTIONS,  
      REG\_EVENTS,  
      SUPPLY\_DAT.  
  
ENDCLASS.  
  
CLASS LCL\_VIEW IMPLEMENTATION.  
  METHOD CONSTRUCTOR.  
    R\_VIEW = ME.  
  ENDMETHOD.  
  METHOD ADD\_LEVEL.  
    DATA: R\_NODES TYPE REF TO CL\_SALV\_NODES,  
          R\_NODE TYPE REF TO CL\_SALV\_NODE,  
          LS\_OUTPUT TYPE \_OUTPUT,  
          LV\_TEXT TYPE LVC\_VALUE.  
    CONSTANTS: C\_PRJ(3) VALUE 'PRJ',  
               C\_WBS(3) VALUE 'WBS'.  
    LS\_OUTPUT = LI\_LINE.  
    R\_NODES = R\_TREE->GET\_NODES( ).  
  
    TRY.  
        R\_NODE = R\_NODES->ADD\_NODE( RELATED\_NODE = LI\_KEY  
                              RELATIONSHIP = CL\_GUI\_COLUMN\_TREE=>RELAT\_LAST\_CHILD ).  
        IF LS\_OUTPUT-LEVEL = 1.  
          LV\_TEXT = LS\_OUTPUT-PROJ\_PSPID.  
\*          CONCATENATE C\_PRJ LS\_OUTPUT-PROJ\_POST1  
\*            INTO LS\_OUTPUT-STEP\_NAME SEPARATED BY SPACE.  
\*          LS\_OUTPUT-STEP\_NAME = LS\_OUTPUT-PROJ\_POST1.  
        ELSEIF LS\_OUTPUT-LEVEL = 2 OR LS\_OUTPUT-LEVEL = 3.  
          LV\_TEXT = LI\_LINE-POSID.  
\*          CONCATENATE C\_WBS LS\_OUTPUT-PRPS\_POST1  
\*            INTO LS\_OUTPUT-STEP\_NAME SEPARATED BY SPACE.  
\*          LS\_OUTPUT-STEP\_NAME = LS\_OUTPUT-PRPS\_POST1.  
        ENDIF.  
        R\_NODE->SET\_TEXT( LV\_TEXT ).  
        R\_NODE->SET\_DATA\_ROW( LS\_OUTPUT ).  
        LO\_KEY = R\_NODE->GET\_KEY( ).  
      CATCH CX\_SALV\_MSG.  
    ENDTRY.  
  ENDMETHOD.  
  METHOD F4\_HLPFILE.  
    CALL FUNCTION 'KD\_GET\_FILENAME\_ON\_F4'  
      EXPORTING  
        PROGRAM\_NAME  = SYST-REPID  
        DYNPRO\_NUMBER = SYST-DYNNR  
        FIELD\_NAME    = EXPATH  
\*       STATIC        = ' '  
\*       MASK          = ' '  
\*       FILEOPERATION = 'R'  
      CHANGING  
        FILE\_NAME     = EXPATH  
\*       LOCATION\_FLAG = 'P'  
      EXCEPTIONS  
        MASK\_TOO\_LONG = 1  
        OTHERS        = 2.  
    IF SY-SUBRC <> 0.  
    ELSE.  
      IF EXPATH IS NOT INITIAL.  
        TRANSLATE EXPATH TO UPPER CASE.                   "#EC SYNTCHAR  
        IF EXPATH CP C\_CSV.  
        ELSE.  
          CONCATENATE EXPATH C\_CSV INTO EXPATH.  
        ENDIF.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.  
  METHOD BUILD\_TREE.  
    EDIT\_COLUMNS( ).  
    GET\_FUNCTIONS( ).  
    CREATE\_METAINF( ).  
    R\_TREE->DISPLAY( ).  
  ENDMETHOD.  
  METHOD CREATE\_CSV\_HDR.  
  
    FIELD-SYMBOLS:  
      <FS\_RESULT> TYPE SOLISTI1.  
  
    APPEND INITIAL LINE TO I\_RESULT ASSIGNING <FS\_RESULT>.  
    CONCATENATE  
      TEXT-C02"'Project'  
      TEXT-C26 "'Project Description'  
      TEXT-C08" 'Task'  
      TEXT-C09"'Task Description'  
      TEXT-C27"'Partner Function'  
      TEXT-C28"'Customer'  
      TEXT-C29"'Department Total Code'  
      TEXT-C11"'Original Cost'  
      TEXT-C13"'Current Budget'  
      TEXT-C17"'Ordered Cost'  
      TEXT-C19"'Actual Cost'  
      TEXT-C15"'Projected End Cost'  
      TEXT-C30"'Division'  
      TEXT-C31"'Snapshot Date'  
      TEXT-C25"Project Status'  
      TEXT-C32"'Task Status'  
      TEXT-C33"'Task Date'  
      INTO <FS\_RESULT>-LINE SEPARATED BY C\_SEP.  
  
  ENDMETHOD.  
  METHOD CREATE\_TREE.  
    TRY.  
        CL\_SALV\_TREE=>FACTORY(  
          IMPORTING  
            R\_SALV\_TREE = R\_TREE  
          CHANGING  
            T\_TABLE      = T\_OUTPUT ).  
      CATCH CX\_SALV\_NO\_NEW\_DATA\_ALLOWED  
            CX\_SALV\_ERROR.  
        EXIT.  
    ENDTRY.  
  ENDMETHOD.  
  METHOD CREATE\_METAINF.  
    DATA: LR\_SETTINGS TYPE REF TO CL\_SALV\_TREE\_SETTINGS,  
          LR\_CONTENT TYPE REF TO CL\_SALV\_FORM\_ELEMENT,  
          LR\_LAYOUT\_LABEL\_GRID TYPE REF TO CL\_SALV\_FORM\_LAYOUT\_DATA\_GRID,  
          LR\_GRID    TYPE REF TO CL\_SALV\_FORM\_LAYOUT\_GRID,  
          LR\_GRID\_1  TYPE REF TO CL\_SALV\_FORM\_LAYOUT\_GRID,  
          LR\_GRID\_2  TYPE REF TO CL\_SALV\_FORM\_LAYOUT\_GRID,  
          LR\_LABEL   TYPE REF TO CL\_SALV\_FORM\_LABEL,  
          LR\_TEXT    TYPE REF TO CL\_SALV\_FORM\_TEXT,  
          LV\_LINES    TYPE I,  
          LV\_ROW      TYPE N,  
          LV\_NUM      TYPE I,  
          LV\_TEXT1     TYPE CHAR20,  
          LS\_OUTPUT  TYPE \_OUTPUT,  
          LS\_TXTPOOL TYPE TEXTPOOL,  
          LT\_TXTPOOL TYPE TABLE OF TEXTPOOL  
          .  
  
    READ TEXTPOOL SY-CPROG INTO LT\_TXTPOOL LANGUAGE SY-LANGU STATE C\_A.  
    LR\_SETTINGS = R\_TREE->GET\_TREE\_SETTINGS( ).  
    LR\_SETTINGS->SET\_HIERARCHY\_HEADER( TEXT-HD1 ).  
    LR\_SETTINGS->SET\_HEADER( TEXT-HD2 ).  
    LR\_SETTINGS->SET\_HIERARCHY\_TOOLTIP( TEXT-HT1 ).  
    LR\_SETTINGS->SET\_HIERARCHY\_SIZE( 30 ).  
    LR\_SETTINGS->SET\_HIERARCHY\_ICON( '@3Q@' ).  
  
  
    READ TABLE T\_OTREE INTO LS\_OUTPUT INDEX 1.  
  
    CREATE OBJECT LR\_GRID.  
    LR\_GRID->CREATE\_HEADER\_INFORMATION( ROW     = 1  
                                        COLUMN  = 2  
                                        TEXT    = V\_COMP\_NAME ).  
    LR\_GRID\_1 = LR\_GRID->CREATE\_GRID( ROW    = 2  
                                      COLUMN = 1 ).  
  
  
    LR\_LABEL = LR\_GRID\_1->CREATE\_LABEL( ROW     = 2  
                                        COLUMN  = 1  
                                        TEXT    = TEXT-H01 ).  
    LR\_LAYOUT\_LABEL\_GRID ?= LR\_LABEL->GET\_LAYOUT\_DATA( ).  
    LR\_LAYOUT\_LABEL\_GRID->SET\_WIDTH( '30' ).  
  
    LR\_GRID\_1->CREATE\_TEXT( ROW     = 2  
                            COLUMN  = 6  
                            TEXT    = ' ' ).  
    LR\_TEXT = LR\_GRID\_1->CREATE\_TEXT( ROW     = 2  
                                       COLUMN  = 2  
                                       TEXT    = SY-REPID  
                                       TOOLTIP = TEXT-H01 ).  
  
    LR\_LABEL = LR\_GRID\_1->CREATE\_LABEL( ROW     = 3  
                                        COLUMN  = 1  
                                        TEXT    = TEXT-H02 ).  
    WRITE SY-DATUM TO LV\_TEXT1 MM/DD/YYYY.  
    WRITE SY-UZEIT TO LV\_TEXT1+11(9) USING EDIT MASK '\_\_:\_\_:\_\_'.  
    LR\_TEXT = LR\_GRID\_1->CREATE\_TEXT( ROW     = 3  
                                      COLUMN  = 2  
                                      TEXT    = LV\_TEXT1  
                                      TOOLTIP = TEXT-H02 ).  
    LR\_LABEL = LR\_GRID\_1->CREATE\_LABEL( ROW     = 4  
                                       COLUMN  = 1  
                                        TEXT    = TEXT-H03 ).  
    LR\_TEXT = LR\_GRID\_1->CREATE\_TEXT( ROW     = 4  
                                      COLUMN  = 2  
                                      TEXT    = SY-UNAME  
                                      TOOLTIP = TEXT-H03 ).  
    LR\_LABEL = LR\_GRID\_1->CREATE\_LABEL( ROW     = 4  
                                        COLUMN  = 3  
                                        TEXT    = TEXT-H04 ).  
    LR\_TEXT = LR\_GRID\_1->CREATE\_TEXT( ROW     = 4  
                                      COLUMN  = 4  
                                      TEXT    = LS\_OUTPUT-PROJ\_PSPID  
                                      TOOLTIP = TEXT-H05 ).  
    LR\_LABEL = LR\_GRID\_1->CREATE\_LABEL( ROW     = 5  
                                        COLUMN  = 1  
                                        TEXT    = TEXT-H06 ).  
    LR\_TEXT = LR\_GRID\_1->CREATE\_TEXT( ROW     = 5  
                                      COLUMN  = 2  
                                      TEXT    = LS\_OUTPUT-PROJ\_TYPE  
                                      TOOLTIP = TEXT-H06 ).  
    DESCRIBE TABLE T\_OUTPUT LINES LV\_LINES.  
    READ TABLE T\_OUTPUT INTO LS\_OUTPUT INDEX LV\_LINES.  
    LR\_LABEL = LR\_GRID\_1->CREATE\_LABEL( ROW     = 5  
                                        COLUMN  = 3  
                                        TEXT    = TEXT-H07  ).  
  
    LR\_TEXT = LR\_GRID\_1->CREATE\_TEXT( ROW     = 5  
                                      COLUMN  = 4  
                                      TEXT    = LS\_OUTPUT-PROJ\_PSPID  
                                      TOOLTIP = TEXT-H05 ).  
  
    LR\_CONTENT = LR\_GRID.  
    R\_TREE->SET\_TOP\_OF\_LIST( LR\_CONTENT ).  
    CLEAR: LR\_LABEL, LR\_TEXT.  
    CREATE OBJECT LR\_GRID\_2.  
    LR\_GRID\_2->CREATE\_GRID(  
      EXPORTING  
        ROW     =     15  
        COLUMN  =     3  
    ).  
    CLEAR: LV\_TEXT1, LV\_LINES.  
    DO 5 TIMES.  
      LV\_ROW = LV\_LINES = LV\_LINES + 1.  
      CONCATENATE 'F0' LV\_ROW INTO LV\_TEXT1.  
      READ TABLE LT\_TXTPOOL INTO LS\_TXTPOOL WITH KEY ID = C\_I KEY = LV\_TEXT1.  
      IF SY-SUBRC = 0.  
        LR\_LABEL = LR\_GRID\_2->CREATE\_LABEL(  
            EXPORTING  
              ROW     =     LV\_LINES  
              COLUMN  =     1  
              TEXT    =    LS\_TXTPOOL-ENTRY  
          ).  
      ENDIF.  
    ENDDO.  
    LV\_LINES = LV\_LINES + 3.  
    DO 4 TIMES.  
      CLEAR LV\_ROW.  
      LV\_LINES = LV\_LINES + 1.  
      CONCATENATE C\_F SY-ABCDE+LV\_NUM(1) LV\_ROW INTO LV\_TEXT1.  
      READ TABLE LT\_TXTPOOL INTO LS\_TXTPOOL WITH KEY ID = C\_I KEY = LV\_TEXT1.  
      IF SY-SUBRC = 0.  
        LR\_LABEL = LR\_GRID\_2->CREATE\_LABEL( ROW     = LV\_LINES  
                                        COLUMN  = 1  
                                        TEXT    = LS\_TXTPOOL-ENTRY ).  
      ENDIF.  
      CLEAR LV\_ROW.  
      DO 3 TIMES.  
        LV\_ROW = LV\_ROW + 1.  
        CONCATENATE C\_F SY-ABCDE+LV\_NUM(1) LV\_ROW INTO LV\_TEXT1.  
        READ TABLE LT\_TXTPOOL INTO LS\_TXTPOOL WITH KEY ID = C\_I KEY = LV\_TEXT1.  
        IF SY-SUBRC = 0.  
          LR\_TEXT = LR\_GRID\_2->CREATE\_TEXT(  
              EXPORTING  
                ROW     =     LV\_LINES  
                COLUMN  =     3  
                TEXT    =    LS\_TXTPOOL-ENTRY  
            ).  
        ENDIF.  
        LV\_LINES = LV\_LINES + 1.  
      ENDDO.  
      LV\_NUM = LV\_NUM + 1.  
    ENDDO.  
    CLEAR LR\_CONTENT.  
    LR\_CONTENT = LR\_GRID\_2.  
    R\_TREE->SET\_END\_OF\_LIST( LR\_CONTENT ).  
  ENDMETHOD.  
  METHOD DISP\_TREE.  
    T\_OTREE[] = T\_OP[].  
    V\_COMP\_NAME = COMP\_NAME.  
    CREATE\_TREE( ).  
    SUPPLY\_DAT( ).  
    REG\_EVENTS( ).  
    BUILD\_TREE( ).  
  
  ENDMETHOD.  
  METHOD EDIT\_COLUMNS.  
    DATA:  
      LR\_COLUMNS TYPE REF TO CL\_SALV\_COLUMNS,  
      LR\_COLUMN  TYPE REF TO CL\_SALV\_COLUMN.  
  
    TRY .  
        LR\_COLUMNS = R\_TREE->GET\_COLUMNS( ).  
        LR\_COLUMNS->SET\_OPTIMIZE( ABAP\_TRUE ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'STEP\_NAME' ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C33 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C33 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C33 ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PROJ\_PLSEZ' ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C02 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C03 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C04 ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'SLS\_PRICE' ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C05 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C05 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C06 ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'DEPT' ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C07 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C07 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C07 ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'ORIG\_COST' ).  
        LR\_COLUMN->SET\_OPTIMIZED( ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C11 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C12 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C12 ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'CURR\_BUDG' ).  
        LR\_COLUMN->SET\_OPTIMIZED( ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C13 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C14 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C14  ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'END\_COST' ).  
        LR\_COLUMN->SET\_OPTIMIZED( ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C15  ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C16  ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C16 ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'ORDR\_COST' ).  
        LR\_COLUMN->SET\_OPTIMIZED( ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C17 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C17 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C18 ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'ACTL\_COST ' ).  
        LR\_COLUMN->SET\_OPTIMIZED( ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C19 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C19  ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C20 ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'ORDR\_PERC' ).  
        LR\_COLUMN->SET\_ALIGNMENT( CL\_SALV\_COLUMN=>RIGHT ).  
        LR\_COLUMN->SET\_OPTIMIZED( ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C21 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C21 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C21 ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'CURR\_ACTL' ).  
        LR\_COLUMN->SET\_OPTIMIZED( ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C22 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C23 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C24 ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'TASK\_STAT' ).  
        LR\_COLUMN->SET\_OPTIMIZED( ).  
        LR\_COLUMN->SET\_OUTPUT\_LENGTH( '6' ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C25 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C25 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C25  ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PROJ\_TYPE' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'COMP\_NAME' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'STATUS' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PROJ\_STAT' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'TASK\_DATE' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'VBUKR' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'VKOKR' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PSPNR' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PRPS\_POSKI' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PRCTR ' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PSPHI' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'OBJNR' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'KUNNR' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PARVW' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PRPS\_USR00' ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C07 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C07 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C07 ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'LEVEL' ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PROJ\_PSPID' ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C01 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C01 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C01 ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PROJ\_POST1' ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-H05 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-H05 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-H05 ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'POSID' ).  
        LR\_COLUMN->SET\_LONG\_TEXT( TEXT-C08 ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C08 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C08 ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
  
        LR\_COLUMN = LR\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PRPS\_POST1' ).  
        LR\_COLUMN->SET\_OPTIMIZED( ).  
        LR\_COLUMN->SET\_LONG\_TEXT(  TEXT-C09  ).  
        LR\_COLUMN->SET\_MEDIUM\_TEXT(  TEXT-C10 ).  
        LR\_COLUMN->SET\_SHORT\_TEXT( TEXT-C10 ).  
        LR\_COLUMN->SET\_VISIBLE( ' ' ).  
      CATCH CX\_SALV\_NOT\_FOUND CX\_SALV\_DATA\_ERROR.  
        EXIT.  
    ENDTRY.  
  ENDMETHOD.  
  METHOD GET\_FUNCTIONS.  
    DATA: R\_FUNCTIONS TYPE REF TO CL\_SALV\_FUNCTIONS\_TREE.  
  
    TRY .  
        R\_FUNCTIONS = R\_TREE->GET\_FUNCTIONS( ).  
      CATCH CX\_SALV\_MSG.  
  
    ENDTRY.  
    R\_FUNCTIONS->SET\_ALL( 'X' ).  
  
  ENDMETHOD.  
  METHOD GET\_PARAMS.  
    DATA LS\_RSDSSELOPT TYPE RSDSSELOPT.  
    IF SY-SUBRC = 0.  
      VBUKR  = P\_VBUKR.  
      VKOKR  = P\_VKOKR.  
      STAT   = P\_STAT.  
      EXPATH = P\_EXPATH.  
      LOOP AT S\_PSPID TRANSPORTING NO FIELDS WHERE LOW IS NOT INITIAL.  
        MOVE-CORRESPONDING S\_PSPID TO LS\_RSDSSELOPT.  
        APPEND LS\_RSDSSELOPT TO PSPID.  
      ENDLOOP.  
      LOOP AT S\_POSID TRANSPORTING NO FIELDS WHERE LOW IS NOT INITIAL..  
        MOVE-CORRESPONDING S\_POSID TO LS\_RSDSSELOPT.  
        APPEND LS\_RSDSSELOPT TO POSID.  
      ENDLOOP.  
      LOOP AT S\_POSKI TRANSPORTING NO FIELDS WHERE LOW IS NOT INITIAL..  
        MOVE-CORRESPONDING S\_POSKI TO LS\_RSDSSELOPT.  
        APPEND LS\_RSDSSELOPT TO POSKI.  
      ENDLOOP.  
      LOOP AT S\_PRCTR TRANSPORTING NO FIELDS WHERE LOW IS NOT INITIAL..  
        MOVE-CORRESPONDING S\_PRCTR TO LS\_RSDSSELOPT.  
        APPEND LS\_RSDSSELOPT TO PRCTR.  
      ENDLOOP.  
      LOOP AT S\_PSPID TRANSPORTING NO FIELDS WHERE LOW IS NOT INITIAL..  
        MOVE-CORRESPONDING S\_PSPID TO LS\_RSDSSELOPT.  
        APPEND LS\_RSDSSELOPT TO PSPID.  
      ENDLOOP.  
      LOOP AT S\_PRART TRANSPORTING NO FIELDS WHERE LOW IS NOT INITIAL..  
        MOVE-CORRESPONDING S\_PRART TO LS\_RSDSSELOPT.  
        APPEND LS\_RSDSSELOPT TO PRART.  
      ENDLOOP.  
  
    ENDIF.  
  ENDMETHOD.  
  METHOD SAVE\_FILE.  
    DATA:  
      LV\_ORIG\_COST     TYPE STRING,  
      LV\_CURR\_BUDG     TYPE STRING,  
      LV\_ORDR\_COST     TYPE STRING,  
      LV\_ACTL\_COST     TYPE STRING,  
      LV\_END\_COST      TYPE STRING,  
      LV\_FNAME TYPE STRING.  
  
    FIELD-SYMBOLS:  
      <FS\_RESULT> TYPE SOLISTI1,  
      <FS\_OUTPUT> TYPE \_OUTPUT.  
    CREATE\_CSV\_HDR( ).  
  
  
    LOOP AT T\_OTREE ASSIGNING <FS\_OUTPUT>.  
      LV\_ORIG\_COST = <FS\_OUTPUT>-ORIG\_COST.  
      LV\_CURR\_BUDG = <FS\_OUTPUT>-CURR\_BUDG.  
  
      LV\_ORDR\_COST = <FS\_OUTPUT>-ORDR\_COST \* 100.  
      CONCATENATE LV\_ORDR\_COST C\_% INTO LV\_ORDR\_COST.  
      LV\_ACTL\_COST = <FS\_OUTPUT>-ACTL\_COST.  
      LV\_END\_COST  = <FS\_OUTPUT>-END\_COST.  
  
      APPEND INITIAL LINE TO I\_RESULT ASSIGNING <FS\_RESULT>.  
      CONCATENATE <FS\_OUTPUT>-PROJ\_PSPID  
                  <FS\_OUTPUT>-PROJ\_POST1  
                  <FS\_OUTPUT>-POSID  
                  <FS\_OUTPUT>-PRPS\_POST1  
                  <FS\_OUTPUT>-PARVW  
                  <FS\_OUTPUT>-KUNNR  
                  <FS\_OUTPUT>-PRPS\_USR00  
                  LV\_ORIG\_COST  
                  LV\_CURR\_BUDG  
                  LV\_ORDR\_COST  
                  LV\_ACTL\_COST  
                  LV\_END\_COST  
                  <FS\_OUTPUT>-PRCTR  
                  SY-DATUM  
                  <FS\_OUTPUT>-PROJ\_STAT  
                  <FS\_OUTPUT>-TASK\_STAT  
                  <FS\_OUTPUT>-TASK\_DATE INTO <FS\_RESULT>-LINE SEPARATED BY C\_SEP.  
  
      CLEAR:  
        LV\_ORIG\_COST,  
        LV\_CURR\_BUDG,  
        LV\_ORDR\_COST,  
        LV\_ACTL\_COST,  
        LV\_END\_COST.  
    ENDLOOP.  
  
    CLEAR LV\_FNAME.  
    LV\_FNAME = P\_EXPATH.  
    IF LV\_FNAME IS NOT INITIAL.  
\*     To Download the Internal Table Data to Specified Location  
      CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>GUI\_DOWNLOAD  
        EXPORTING  
          FILENAME                  = LV\_FNAME  
          FILETYPE                  = 'ASC'  
\*         APPEND                    = 'X'  
          WRITE\_FIELD\_SEPARATOR     = 'X'  
\*         HEADER                    = '00'  
\*         TRUNC\_TRAILING\_BLANKS     = SPACE  
\*         WRITE\_LF                  = 'X'  
\*         COL\_SELECT                = SPACE  
\*         COL\_SELECT\_MASK           = SPACE  
\*         DAT\_MODE                  = SPACE  
          CONFIRM\_OVERWRITE         = 'X'   "SPACE  
\*         NO\_AUTH\_CHECK             = SPACE  
\*         CODEPAGE                  = SPACE  
\*         IGNORE\_CERR               = ABAP\_TRUE  
\*         REPLACEMENT               = '#'  
\*         WRITE\_BOM                 = SPACE  
          TRUNC\_TRAILING\_BLANKS\_EOL = ''  
\*         WK1\_N\_FORMAT              = SPACE  
\*         WK1\_N\_SIZE                = SPACE  
\*         WK1\_T\_FORMAT              = SPACE  
\*         WK1\_T\_SIZE                = SPACE  
\*        IMPORTING  
\*         FILELENGTH                =  
        CHANGING  
          DATA\_TAB                  = I\_RESULT  
        EXCEPTIONS  
          FILE\_WRITE\_ERROR          = 1  
          NO\_BATCH                  = 2  
          GUI\_REFUSE\_FILETRANSFER   = 3  
          INVALID\_TYPE              = 4  
          NO\_AUTHORITY              = 5  
          UNKNOWN\_ERROR             = 6  
          HEADER\_NOT\_ALLOWED        = 7  
          SEPARATOR\_NOT\_ALLOWED     = 8  
          FILESIZE\_NOT\_ALLOWED      = 9  
          HEADER\_TOO\_LONG           = 10  
          DP\_ERROR\_CREATE           = 11  
          DP\_ERROR\_SEND             = 12  
          DP\_ERROR\_WRITE            = 13  
          UNKNOWN\_DP\_ERROR          = 14  
          ACCESS\_DENIED             = 15  
          DP\_OUT\_OF\_MEMORY          = 16  
          DISK\_FULL                 = 17  
          DP\_TIMEOUT                = 18  
          FILE\_NOT\_FOUND            = 19  
          DATAPROVIDER\_EXCEPTION    = 20  
          CONTROL\_FLUSH\_ERROR       = 21  
          NOT\_SUPPORTED\_BY\_GUI      = 22  
          ERROR\_NO\_GUI              = 23  
          OTHERS                    = 24.  
      IF SY-SUBRC <> 0.  
        LCL\_LOG\_UTILITY=>POPULATE\_LOG( EXPORTING  
                                         I\_TYPE   = C\_I  
                                         I\_CL     = C\_MESS\_CL  
                                         I\_NUMBER = 003  
                                         I\_PAR1   = SY-MSGV1  
                                         I\_PAR2   = SY-MSGV2  
                                         I\_PAR3   = SY-MSGV3  
                                         I\_PAR4   = SY-MSGV4  
                                       CHANGING  
                                         C\_LOG    = T\_LOG ).  
        MESSAGE  S003(ZDW\_DEV\_OU) WITH 'File Successfully'(010) 'Downloaded to :'(011) LV\_FNAME .  
      ENDIF.  
    ELSE.  
      MESSAGE  S001(ZDW\_DEV\_OU) WITH 'File not found'(012).  
    ENDIF.  
  ENDMETHOD.  
  METHOD SUPPLY\_DAT.  
  
    DATA: LV\_PROJ\_PSPID TYPE PS\_PSPID,  
          LV\_POSID TYPE PS\_POSID,  
          LV\_PRPS\_USR00 TYPE USR00PRPS,  
          LV\_KEY1 TYPE LVC\_NKEY,  
          LV\_KEY2 TYPE LVC\_NKEY,  
          LV\_KEY3 TYPE LVC\_NKEY,  
          LV\_KEY4 TYPE LVC\_NKEY,  
          LS\_OUTPUT TYPE \_OUTPUT.  
    LOOP AT T\_OTREE INTO LS\_OUTPUT.  
  
      IF LS\_OUTPUT-LEVEL = 1.  
        CLEAR: LV\_KEY1, LV\_KEY2.  
        ADD\_LEVEL( EXPORTING LI\_KEY = SPACE  
                     LI\_LINE = LS\_OUTPUT  
           IMPORTING LO\_KEY = LV\_KEY1 ).  
        CONTINUE.  
      ELSEIF LS\_OUTPUT-LEVEL = 2.  
        ADD\_LEVEL( EXPORTING LI\_KEY = LV\_KEY1  
                     LI\_LINE = LS\_OUTPUT  
           IMPORTING LO\_KEY = LV\_KEY2 ).  
        CONTINUE.  
      ELSEIF LS\_OUTPUT-LEVEL = 3.  
        ADD\_LEVEL( EXPORTING LI\_KEY = LV\_KEY2  
                     LI\_LINE = LS\_OUTPUT  
           IMPORTING LO\_KEY = LV\_KEY3 ).  
        CONTINUE.  
      ENDIF.  
  
    ENDLOOP.  
  ENDMETHOD.  
  METHOD CHECK\_FILE.  
    CONSTANTS:  
      C\_CSV2(3)    VALUE 'CSV'.  
    DATA:  
      LV\_DIR     TYPE STRING,     " For Directory  
      LV\_BOL     TYPE ABAP\_BOOL,  " Result  
      LV\_FNAME   TYPE STRING,     " File Name  
      LV\_FNAME1  TYPE STRING,     " File Name  
      LV\_ERR\_MESS TYPE SYMSGV,  
      LV\_EXT     TYPE STRING.     " File Extension  
  
  
    CALL FUNCTION 'SO\_SPLIT\_FILE\_AND\_PATH'  
      EXPORTING  
        FULL\_NAME     = P\_EXPATH  
      IMPORTING  
        STRIPPED\_NAME = LV\_FNAME  
        FILE\_PATH     = LV\_DIR  
      EXCEPTIONS  
        X\_ERROR       = 1  
        OTHERS        = 2.  
    IF SY-SUBRC <> 0.  
      LCL\_LOG\_UTILITY=>POPULATE\_LOG( EXPORTING  
                                       I\_TYPE   = C\_I  
                                       I\_CL     = C\_MESS\_CL  
                                       I\_NUMBER = 003  
                                       I\_PAR1   = SY-MSGV1  
                                       I\_PAR2   = SY-MSGV2  
                                       I\_PAR3   = SY-MSGV3  
                                       I\_PAR4   = SY-MSGV4  
                                     CHANGING  
                                       C\_LOG    = T\_LOG ).  
    ENDIF.  
  
    CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>DIRECTORY\_EXIST  
      EXPORTING  
        DIRECTORY            = LV\_DIR  
      RECEIVING  
        RESULT               = LV\_BOL  
      EXCEPTIONS  
        CNTL\_ERROR           = 1  
        ERROR\_NO\_GUI         = 2  
        WRONG\_PARAMETER      = 3  
        NOT\_SUPPORTED\_BY\_GUI = 4  
        OTHERS               = 5.  
    IF SY-SUBRC = 0.  
    ELSE.  
      IF LV\_BOL IS INITIAL.  
        LV\_ERR\_MESS = LV\_DIR.  
        LCL\_LOG\_UTILITY=>POPULATE\_LOG( EXPORTING  
                                         I\_TYPE   = C\_E  
                                         I\_CL     = C\_MESS\_CL  
                                         I\_NUMBER = 015  
                                         I\_PAR1   = LV\_ERR\_MESS  
                                       CHANGING  
                                         C\_LOG    = T\_LOG ).  
      ELSE.  
        TRANSLATE LV\_FNAME TO UPPER CASE.                 "#EC SYNTCHAR  
        SPLIT LV\_FNAME AT '.' INTO LV\_FNAME1 LV\_EXT.  
        IF LV\_EXT IS INITIAL.  
          LV\_ERR\_MESS = LV\_DIR.  
          LCL\_LOG\_UTILITY=>POPULATE\_LOG( EXPORTING  
                                           I\_TYPE   = C\_E  
                                           I\_CL     = C\_MESS\_CL  
                                           I\_NUMBER = 016  
                                         CHANGING  
                                           C\_LOG    = T\_LOG ).  
        ELSE.  
          IF LV\_EXT CP C\_CSV OR LV\_EXT EQ C\_CSV2.  
  
          ELSE.  
            LV\_ERR\_MESS = LV\_EXT.  
            LCL\_LOG\_UTILITY=>POPULATE\_LOG( EXPORTING  
                                             I\_TYPE   = C\_E  
                                             I\_CL     = C\_MESS\_CL  
                                             I\_NUMBER = 017  
                                             I\_PAR1   = LV\_ERR\_MESS  
                                           CHANGING  
                                             C\_LOG    = T\_LOG ).  
  
          ENDIF.  
        ENDIF.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.  
  METHOD WRITE\_LOG.  
  
    CONSTANTS:  
      C\_NO\_DATA(29)    VALUE 'No valid data records to load'.  
  
    DATA:LV\_RECORDS TYPE I,  
         LS\_RETURN  TYPE BAPIRET2.  
  
    IF T\_LOG IS INITIAL AND T\_OUTPUT[] IS INITIAL.  
      LS\_RETURN-TYPE = C\_E.  
      LS\_RETURN-MESSAGE = C\_NO\_DATA.  
      APPEND LS\_RETURN TO T\_LOG.  
    ENDIF.  
    LV\_RECORDS = LINES(  T\_OUTPUT[] ).  
    LCL\_LOG\_UTILITY=>WRITE\_LOG( EXPORTING  
                                  I\_PROG\_LOG  = T\_LOG  
                                  I\_RECS      = LV\_RECORDS  
                                  I\_FILENAME1 = EXPATH ).  
  
  
  ENDMETHOD.  
  METHOD REG\_EVENTS.  
\*... §4.3 register to the events of cl\_salv\_table  
    DATA: LR\_EVENTS TYPE REF TO CL\_SALV\_EVENTS\_TREE.  
  
    LR\_EVENTS = R\_TREE->GET\_EVENT( ).  
  
    CREATE OBJECT R\_EVENTS.  
  
    SET HANDLER R\_EVENTS->ON\_DOUBLE\_CLICK FOR LR\_EVENTS.  
  
    SET HANDLER R\_EVENTS->ON\_LINK\_CLICK FOR LR\_EVENTS.  
  
    SET HANDLER R\_EVENTS->ON\_BEFORE\_USER\_COMMAND FOR LR\_EVENTS.  
  
    SET HANDLER R\_EVENTS->ON\_AFTER\_USER\_COMMAND FOR LR\_EVENTS.  
  
    SET HANDLER R\_EVENTS->ON\_KEYPRESS FOR LR\_EVENTS.  
  
\* register the keys for which keypress should be raised  
    TRY.  
        LR\_EVENTS->ADD\_KEY\_FOR\_KEYPRESS( IF\_SALV\_C\_KEYS=>F1 ).  
        LR\_EVENTS->ADD\_KEY\_FOR\_KEYPRESS( IF\_SALV\_C\_KEYS=>F4 ).  
        LR\_EVENTS->ADD\_KEY\_FOR\_KEYPRESS( IF\_SALV\_C\_KEYS=>ENTER ).  
      CATCH CX\_SALV\_MSG.  
    ENDTRY.  
    R\_EVENTS->T\_OTREE = ME->T\_OTREE.  
  ENDMETHOD.  
ENDCLASS.  
CLASS LCL\_DB\_IO DEFINITION.  
  PUBLIC SECTION.  
    METHODS CONSTRUCTOR.  
    DATA:  
          T\_PROJ  TYPE TABLE OF \_PROJ,  
          T\_PRPS  TYPE TABLE OF \_PRPS,  
          T\_PSTAT TYPE TABLE OF JEST,  
          T\_TSTAT TYPE TABLE OF JEST,  
          T\_TPART TYPE TABLE OF TPART,  
          T\_KNA1  TYPE TABLE OF KNA1,  
          T\_VBAP  TYPE TABLE OF VBAP,  
          T\_VBPA  TYPE TABLE OF VBPA,  
          T\_RPSCO TYPE TABLE OF \_RPSCO,  
          T\_TJ02T TYPE TABLE OF TJ02T,  
          T\_COEP  TYPE TABLE OF COEP,  
          T\_BPGE  TYPE TABLE OF BPGE,  
          T\_BPHI  TYPE TABLE OF BPHI,  
          T\_TCJ1T TYPE TABLE OF TCJ1T,  
          T\_COOI  TYPE TABLE OF COOI.  
    DATA:  
          VBUKR TYPE PROJ-VBUKR,  
          VKOKR TYPE PROJ-VKOKR,  
          STAT  TYPE JEST-STAT,  
          PSPID TYPE RSELOPTION,  
          POSID TYPE RSELOPTION,  
          POSKI TYPE RSELOPTION,  
          PRCTR TYPE RSELOPTION,  
          PRART TYPE RSELOPTION,  
          COMP\_NAME TYPE T880-NAME1,  
          EXPATH TYPE RLGRAP-FILENAME.  
ENDCLASS.  
CLASS LCL\_DB\_IO IMPLEMENTATION.  
  METHOD CONSTRUCTOR.  
    CLEAR:T\_PROJ[],  
          T\_PRPS[],  
          T\_PSTAT[],  
          T\_TSTAT[],  
          T\_TPART[],  
          T\_KNA1[],  
          T\_VBAP[],  
          T\_VBPA[],  
          T\_RPSCO[],  
          T\_TJ02T[],  
          T\_COEP[],  
          T\_BPGE[],  
          T\_TCJ1T[],  
          T\_COOI[],  
          VBUKR,  
          VKOKR,  
          STAT,  
          PSPID,  
          POSID,  
          POSKI,  
          PRCTR,  
          PRART.  
  ENDMETHOD.  
ENDCLASS.  
CLASS LCL\_DBAPI DEFINITION.  
  PUBLIC SECTION.  
    METHODS:  
     FETCH\_DATA CHANGING RC\_DATA TYPE REF TO LCL\_DB\_IO.  
ENDCLASS.  
  
CLASS LCL\_DBAPI IMPLEMENTATION.  
  METHOD FETCH\_DATA.  
  
    DATA:  
      LV\_COMPCODE    TYPE RCOMP\_D.  
  
    UNPACK RC\_DATA->VBUKR TO LV\_COMPCODE.  
    SELECT SINGLE NAME1 INTO RC\_DATA->COMP\_NAME FROM T880  
      WHERE RCOMP = LV\_COMPCODE.  
  
    IF RC\_DATA->PSPID[] IS NOT INITIAL.  
      SELECT VBUKR VKOKR PSPNR PSPID POST1 PLSEZ OBJNR FROM PROJ  
        INTO CORRESPONDING FIELDS OF TABLE RC\_DATA->T\_PROJ  
        WHERE PSPID IN RC\_DATA->PSPID AND  
              VBUKR  = RC\_DATA->VBUKR AND  
              VKOKR  = RC\_DATA->VKOKR.  
  
      SELECT POSID POSKI PRCTR PRART POST1 PSPHI OBJNR USR00 PSPNR ERDAT STUFE FROM PRPS  
        INTO CORRESPONDING FIELDS OF TABLE RC\_DATA->T\_PRPS  
        FOR ALL ENTRIES IN RC\_DATA->T\_PROJ  
        WHERE PSPHI  = RC\_DATA->T\_PROJ-PSPNR AND  
              PRCTR IN RC\_DATA->PRCTR      AND  
              PRART IN RC\_DATA->PRART      AND  
              POSKI IN RC\_DATA->POSKI.  
    ELSE.  
      IF RC\_DATA->POSID[] IS NOT INITIAL.  
        SELECT POSID POSKI PRCTR PRART POST1 PSPHI OBJNR USR00 PSPNR ERDAT STUFE  
          FROM PRPS  
          INTO CORRESPONDING FIELDS OF TABLE RC\_DATA->T\_PRPS  
          WHERE POSID IN RC\_DATA->POSID  AND  
                PRCTR IN RC\_DATA->PRCTR  AND  
                PRART IN RC\_DATA->PRART  AND  
                POSKI IN RC\_DATA->POSKI.  
  
        SELECT VBUKR VKOKR PSPNR PSPID POST1 PLSEZ OBJNR  
          FROM PROJ  
          INTO CORRESPONDING FIELDS OF TABLE RC\_DATA->T\_PROJ  
          FOR ALL ENTRIES IN RC\_DATA->T\_PRPS  
          WHERE PSPNR  = RC\_DATA->T\_PRPS-PSPHI AND  
                VBUKR  = RC\_DATA->VBUKR      AND  
                VKOKR  = RC\_DATA->VKOKR.  
  
        SELECT POSID POSKI PRCTR PRART POST1 PSPHI OBJNR USR00 PSPNR ERDAT STUFE  
          FROM PRPS  
          INTO CORRESPONDING FIELDS OF TABLE RC\_DATA->T\_PRPS  
          FOR ALL ENTRIES IN RC\_DATA->T\_PROJ  
          WHERE PSPHI  = RC\_DATA->T\_PROJ-PSPNR AND  
                PRCTR IN RC\_DATA->PRCTR      AND  
                PRART IN RC\_DATA->PRART      AND  
                POSKI IN RC\_DATA->POSKI.  
      ELSE.  
        IF RC\_DATA->POSKI IS NOT INITIAL.  
          SELECT POSID POSKI PRCTR PRART POST1 PSPHI OBJNR USR00 PSPNR ERDAT STUFE  
            FROM PRPS  
            INTO CORRESPONDING FIELDS OF TABLE RC\_DATA->T\_PRPS  
            WHERE POSKI IN RC\_DATA->POSKI  AND  
                  PRCTR IN RC\_DATA->PRCTR  AND  
                  PRART IN RC\_DATA->PRART.  
  
          SELECT VBUKR VKOKR PSPNR PSPID POST1 PLSEZ OBJNR FROM PROJ  
            INTO CORRESPONDING FIELDS OF TABLE RC\_DATA->T\_PROJ  
            FOR ALL ENTRIES IN RC\_DATA->T\_PRPS  
            WHERE PSPNR  = RC\_DATA->T\_PRPS-PSPHI AND  
                  VBUKR  = RC\_DATA->VBUKR      AND  
                  VKOKR  = RC\_DATA->VKOKR.  
  
          SELECT POSID POSKI PRCTR PRART POST1 PSPHI OBJNR USR00 PSPNR ERDAT STUFE  
            FROM PRPS  
            INTO CORRESPONDING FIELDS OF TABLE RC\_DATA->T\_PRPS  
            FOR ALL ENTRIES IN RC\_DATA->T\_PROJ  
            WHERE PSPHI  = RC\_DATA->T\_PROJ-PSPNR AND  
                  PRCTR IN RC\_DATA->PRCTR      AND  
                  PRART IN RC\_DATA->PRART      AND  
                  POSKI IN RC\_DATA->POSKI.  
        ENDIF.  
      ENDIF.  
    ENDIF.  
  
    IF RC\_DATA->T\_PROJ IS NOT INITIAL.  
      SELECT \* FROM JEST  
        INTO TABLE RC\_DATA->T\_PSTAT  
        FOR ALL ENTRIES IN RC\_DATA->T\_PROJ  
        WHERE OBJNR = RC\_DATA->T\_PROJ-OBJNR  
        .  
    ENDIF.  
  
    IF RC\_DATA->T\_PRPS IS NOT INITIAL.  
      SELECT \* FROM JEST  
        INTO TABLE RC\_DATA->T\_TSTAT  
        FOR ALL ENTRIES IN RC\_DATA->T\_PRPS  
        WHERE OBJNR = RC\_DATA->T\_PRPS-OBJNR  
        .  
  
      SELECT OBJNR WRTTP GJAHR VORGA VERSN BELTP WLP00  
        FROM RPSCO  
        INTO TABLE RC\_DATA->T\_RPSCO  
        FOR ALL ENTRIES IN RC\_DATA->T\_PRPS  
        WHERE OBJNR = RC\_DATA->T\_PRPS-OBJNR.  
  
      SELECT \*  
        FROM TPART  
        INTO TABLE RC\_DATA->T\_TPART  
        ORDER BY SPRAS PARVW.  
  
      SELECT \*  
        FROM VBAP  
        INTO TABLE RC\_DATA->T\_VBAP  
        FOR ALL ENTRIES IN RC\_DATA->T\_PRPS  
        WHERE PS\_PSP\_PNR = RC\_DATA->T\_PRPS-PSPNR.  
  
      SELECT \*  
        FROM VBPA  
        INTO TABLE RC\_DATA->T\_VBPA  
        FOR ALL ENTRIES IN RC\_DATA->T\_VBAP  
        WHERE VBELN = RC\_DATA->T\_VBAP-VBELN.  
  
      SELECT \*  
        FROM KNA1  
        INTO TABLE RC\_DATA->T\_KNA1.  
  
      SELECT \* FROM TJ02T INTO TABLE RC\_DATA->T\_TJ02T.  
  
      SELECT \*  
        FROM COEP  
        INTO TABLE RC\_DATA->T\_COEP  
        FOR ALL ENTRIES IN RC\_DATA->T\_PRPS  
        WHERE OBJNR = RC\_DATA->T\_PRPS-OBJNR AND  
              KOKRS = P\_VKOKR  
        .  
  
      SELECT \*  
        FROM BPGE  
        INTO TABLE RC\_DATA->T\_BPGE  
        FOR ALL ENTRIES IN RC\_DATA->T\_PRPS  
        WHERE OBJNR = RC\_DATA->T\_PRPS-OBJNR.  
  
      SELECT \*  
        FROM COOI  
        INTO TABLE RC\_DATA->T\_COOI  
        FOR ALL ENTRIES IN RC\_DATA->T\_PRPS  
        WHERE OBJNR = RC\_DATA->T\_PRPS-OBJNR  
        .  
  
      SELECT \*  
        FROM TCJ1T  
        INTO TABLE RC\_DATA->T\_TCJ1T  
        FOR ALL ENTRIES IN RC\_DATA->T\_PRPS  
        WHERE PRART = RC\_DATA->T\_PRPS-PRART.  
  
      SELECT \* FROM BPHI  
        INTO TABLE RC\_DATA->T\_BPHI  
       FOR ALL ENTRIES IN RC\_DATA->T\_PRPS  
       WHERE OBJNR = RC\_DATA->T\_PRPS-OBJNR AND  
             WRTTP = C\_01.  
    ENDIF.  
  ENDMETHOD.  
ENDCLASS.  
  
CLASS LCL\_MODEL DEFINITION.  
  PUBLIC SECTION.  
    CLASS-DATA:  
          R\_MODEL TYPE REF TO LCL\_MODEL.  
    METHODS:  
      CONSTRUCTOR,  
      ACCESS\_DATA      RETURNING VALUE(RVAL) TYPE BOOLEAN,  
      TBL\_CONT\_MSGS,  
      SORT\_TABLE,  
      FORMAT\_DATA,  
      CHECK\_FOR\_ERRORS IMPORTING VALUE(T\_ERRTAB) TYPE BAPIRET2\_T  
                       RETURNING VALUE(RVAL) TYPE BOOLEAN,  
      FILTER\_WBS\_ELEMS IMPORTING I\_STR1 TYPE ANY  
                                 I\_STR2 TYPE ANY  
                       RETURNING VALUE(RVAL) TYPE BOOLEAN,  
      GET\_CUSTDATA     IMPORTING I\_STR1  TYPE ANY  
                       EXPORTING E\_KUNNR TYPE NAME1\_GP  
                                 E\_PARVW TYPE VTXTK,  
      GET\_STATTXT       IMPORTING I\_TASK TYPE J\_OBJNR OPTIONAL  
                                  I\_PROJ TYPE J\_OBJNR OPTIONAL  
                        RETURNING VALUE(RVAL) TYPE J\_STEXT,  
      GET\_ACTUALS       IMPORTING I\_VAR  TYPE ANY  
                        CHANGING  I\_STR1 TYPE ANY,  
  
      GET\_ORDERED       IMPORTING I\_VAR  TYPE ANY  
                        CHANGING  I\_STR1 TYPE ANY,  
  
      GET\_VERSION       IMPORTING I\_STR1 TYPE ANY  
                        RETURNING VALUE(RVAL) TYPE I,  
      CONV\_ELEMENTS     CHANGING STR1 TYPE ANY,  
      CHECK\_VALUES     IMPORTING I\_STR TYPE ANY  
                       RETURNING VALUE(RVAL) TYPE BOOLEAN.  
    DATA:  
        R\_DB\_IO TYPE REF TO LCL\_DB\_IO,  
        R\_PERSIST\_DB TYPE REF TO LCL\_DBAPI,  
        T\_LOG TYPE TABLE OF BAPIRET2,  
        V\_ERROR\_CHECK,  
        V\_RECORDS TYPE I,  
        T\_OUTPUT TYPE TABLE OF \_OUTPUT,  
        T\_OPTREE TYPE TABLE OF \_OUTPUT.  
  PRIVATE SECTION.  
    METHODS:  
      DB\_CALL CHANGING RC\_DATA TYPE REF TO LCL\_DB\_IO.  
ENDCLASS.  
CLASS LCL\_MODEL IMPLEMENTATION.  
  METHOD CONSTRUCTOR.  
    CREATE OBJECT R\_DB\_IO.  
    R\_MODEL = ME.  
  ENDMETHOD.  
  METHOD CHECK\_FOR\_ERRORS.  
  
    FIELD-SYMBOLS:  
      <FS\_ILOG> TYPE BAPIRET2.  
  
    CLEAR RVAL.  
    LOOP AT T\_ERRTAB ASSIGNING <FS\_ILOG>.  
      IF <FS\_ILOG>-TYPE = C\_E.  
        RVAL = ABAP\_TRUE.  
        RETURN.  
      ENDIF.  
    ENDLOOP.  
  
  ENDMETHOD.  
  METHOD ACCESS\_DATA.  
    DB\_CALL( CHANGING RC\_DATA = R\_DB\_IO ).  
    SORT\_TABLE(  ).  
    RVAL = CHECK\_FOR\_ERRORS( EXPORTING T\_ERRTAB = T\_LOG ).  
  ENDMETHOD.  
  METHOD CONV\_ELEMENTS.  
  
    FIELD-SYMBOLS:  
      <STR>  TYPE \_OUTPUT.  
  
    ASSIGN STR1 TO <STR>.  
    CALL FUNCTION 'CONVERSION\_EXIT\_PROJN\_OUTPUT'  
      EXPORTING  
        INPUT  = <STR>-PROJ\_PSPID  
      IMPORTING  
        OUTPUT = <STR>-PROJ\_PSPID.  
  
    CALL FUNCTION 'CONVERSION\_EXIT\_PROJN\_OUTPUT'  
      EXPORTING  
        INPUT  = <STR>-POSID  
      IMPORTING  
        OUTPUT = <STR>-POSID.  
  
  ENDMETHOD.  
  METHOD FILTER\_WBS\_ELEMS.  
  
    DATA:  
      V\_POS1  TYPE STRING,  
      V\_POS2  TYPE STRING,  
      V\_POS3  TYPE STRING,  
      V\_WBS   TYPE PS\_POSID,  
      V\_WLP00 TYPE BP\_WPL,  
      V\_CONTR TYPE I,  
      LV\_VERSION       TYPE BP\_VERSION,  
      S\_STR   TYPE \_PRPS.  
  
    FIELD-SYMBOLS:  
      <FS\_RPSCO> TYPE \_RPSCO,  
      <FS\_TASK>  LIKE LINE OF ME->R\_DB\_IO->T\_TSTAT.  
  
    S\_STR = I\_STR1.  
    CALL FUNCTION 'CONVERSION\_EXIT\_ABPSP\_OUTPUT'  
      EXPORTING  
        INPUT  = S\_STR-PSPNR  
      IMPORTING  
        OUTPUT = V\_WBS.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*WBS element A.XXXXXXX.002.XXX or E.XXXXXXX.002.XXX  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
    SPLIT V\_WBS AT C\_PERIOD INTO V\_POS1 V\_POS2.  
    IF V\_POS1 = C\_E OR V\_POS1 = C\_A.  
      CLEAR:  
        V\_POS1.  
      SPLIT V\_POS2 AT C\_PERIOD INTO V\_POS1 V\_POS3.  
      IF V\_POS3(3) = C\_002.  
  
\*        V\_CONTR = V\_CONTR + 1.  
\*\*\*Check Status  
  
        READ TABLE ME->R\_DB\_IO->T\_TSTAT ASSIGNING <FS\_TASK> WITH KEY OBJNR = S\_STR-OBJNR  
                                                                     BINARY SEARCH.  
        IF SY-SUBRC = 0.  
          IF ( <FS\_TASK>-STAT = C\_I0045 AND <FS\_TASK>-INACT = C\_X ) OR  
             ( <FS\_TASK>-STAT = C\_I0046 AND <FS\_TASK>-INACT = C\_X ).  
            RVAL = ABAP\_FALSE.  
          ELSE.  
            RVAL = ABAP\_TRUE.  
          ENDIF.  
        ELSE.  
          RVAL = ABAP\_FALSE.  
        ENDIF.  
\*        IF V\_CONTR IS INITIAL.  
\*          RVAL = ABAP\_FALSE.  
\*        ENDIF.  
      ELSE.  
        RVAL = ABAP\_FALSE.  
      ENDIF.  
    ELSE.  
      RVAL = ABAP\_FALSE.  
    ENDIF.  
  
  ENDMETHOD.  
  METHOD GET\_CUSTDATA.  
  
    DATA:  
      S\_STR   TYPE \_PRPS.  
    FIELD-SYMBOLS:  
      <TPART> TYPE TPART,  
      <KNA1>  TYPE KNA1,  
      <VBAP>  TYPE VBAP,  
      <VBPA>  TYPE VBPA.  
  
    S\_STR = I\_STR1.  
    READ TABLE ME->R\_DB\_IO->T\_VBAP  
      ASSIGNING <VBAP>  
      WITH KEY PS\_PSP\_PNR = S\_STR-PSPNR  
      BINARY SEARCH.  
    IF SY-SUBRC = 0.  
      READ TABLE ME->R\_DB\_IO->T\_VBPA  
        ASSIGNING <VBPA>  
        WITH KEY VBELN = <VBAP>-VBELN  
        BINARY SEARCH.  
      IF SY-SUBRC = 0.  
        READ TABLE ME->R\_DB\_IO->T\_TPART  
          ASSIGNING <TPART> WITH KEY SPRAS = SY-LANGU  
                                     PARVW = <VBPA>-PARVW  
          BINARY SEARCH.  
        IF SY-SUBRC = 0.  
          E\_PARVW = <TPART>-VTEXT.  
        ENDIF.  
        READ TABLE ME->R\_DB\_IO->T\_KNA1  
          ASSIGNING <KNA1> WITH KEY KUNNR = <VBPA>-KUNNR  
          BINARY SEARCH.  
        IF SY-SUBRC = 0.  
          E\_KUNNR = <KNA1>-NAME1.  
        ENDIF.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.  
  METHOD GET\_ACTUALS.  
  
    DATA:  
      LV\_OBJNR TYPE J\_OBJNR.  
  
    FIELD-SYMBOLS:  
      <S\_STR>       TYPE \_OUTPUT,  
      <COEP>        TYPE COEP.  
  
    LV\_OBJNR = I\_VAR.  
    ASSIGN I\_STR1 TO <S\_STR>.  
    LOOP AT ME->R\_DB\_IO->T\_COEP  
         ASSIGNING <COEP>  
         WHERE OBJNR = LV\_OBJNR.  
      <S\_STR>-ACTL\_COST = <S\_STR>-ACTL\_COST + <COEP>-WTGBTR.  
    ENDLOOP.  
  
  
  ENDMETHOD.  
  METHOD GET\_ORDERED.  
    DATA:  
      LV\_OBJNR TYPE J\_OBJNR.  
  
    FIELD-SYMBOLS:  
      <S\_STR>       TYPE \_OUTPUT,  
      <BPGE>        TYPE BPGE,  
      <COOI>        TYPE COOI.  
  
    LV\_OBJNR = I\_VAR.  
    ASSIGN I\_STR1 TO <S\_STR>.  
    LOOP AT ME->R\_DB\_IO->T\_COOI ASSIGNING <COOI> WHERE OBJNR = LV\_OBJNR.  
      <S\_STR>-ORDR\_COST = <S\_STR>-ORDR\_COST + <COOI>-WHGBTR.  
    ENDLOOP.  
    <S\_STR>-ORDR\_COST = <S\_STR>-ORDR\_COST + <S\_STR>-ACTL\_COST.  
  
  ENDMETHOD.  
  METHOD GET\_STATTXT.  
    FIELD-SYMBOLS:  
      <FS\_TASK>     TYPE JEST,  
      <TJ02T>    TYPE TJ02T.  
  
    DATA:  
      LV\_ANW\_STAT\_EXISTING TYPE  XFELD,  
      LV\_E\_STSMA TYPE J\_STSMA,  
      LV\_LINE    TYPE J\_STEXT,  
      LV\_STAT  TYPE J\_STATUS,  
      LV\_STONR   TYPE J\_STONR.  
  
  
    IF I\_TASK IS NOT INITIAL.  
      READ TABLE ME->R\_DB\_IO->T\_TSTAT ASSIGNING <FS\_TASK> WITH KEY OBJNR = I\_TASK  
                                                    STAT  = C\_I0043 BINARY SEARCH.  
      IF SY-SUBRC = 0 AND <FS\_TASK>-INACT IS INITIAL.  
        LV\_STAT = C\_I0043.  
      ELSE.  
        READ TABLE ME->R\_DB\_IO->T\_TSTAT ASSIGNING <FS\_TASK> WITH KEY OBJNR = I\_TASK  
                                                    STAT  = C\_I0067 BINARY SEARCH.  
        IF SY-SUBRC = 0 AND <FS\_TASK>-INACT IS INITIAL.  
          LV\_STAT = C\_I0067.  
        ELSE.  
          CALL FUNCTION 'STATUS\_TEXT\_EDIT'  
            EXPORTING  
              OBJNR             = I\_TASK  
              SPRAS             = SY-LANGU  
            IMPORTING  
              ANW\_STAT\_EXISTING = LV\_ANW\_STAT\_EXISTING  
              E\_STSMA           = LV\_E\_STSMA  
              LINE              = LV\_LINE  
              USER\_LINE         = RVAL  
              STONR             = LV\_STONR.  
          IF RVAL IS INITIAL.  
            RVAL = LV\_LINE.  
          ENDIF.  
        ENDIF.  
      ENDIF.  
    ELSEIF I\_PROJ IS NOT INITIAL.  
      CALL FUNCTION 'STATUS\_TEXT\_EDIT'  
        EXPORTING  
          OBJNR             = I\_PROJ  
          SPRAS             = SY-LANGU  
        IMPORTING  
          ANW\_STAT\_EXISTING = LV\_ANW\_STAT\_EXISTING  
          E\_STSMA           = LV\_E\_STSMA  
          LINE              = LV\_LINE  
          USER\_LINE         = RVAL  
          STONR             = LV\_STONR.  
    ELSE.  
      READ TABLE ME->R\_DB\_IO->T\_TJ02T ASSIGNING <TJ02T> WITH KEY ISTAT = LV\_STAT  
                                                    SPRAS = SY-LANGU BINARY SEARCH.  
      IF SY-SUBRC = 0.  
        RVAL = <TJ02T>-TXT30.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.  
  METHOD GET\_VERSION.  
  
    DATA:  
      LV\_DTYP(4),  
      LS\_STR   TYPE \_PROJ.  
  
    FIELD-SYMBOLS:  
      <FS\_BPHI>  TYPE BPHI.  
  
    LS\_STR = I\_STR1.  
  
    LOOP AT  ME->R\_DB\_IO->T\_BPHI ASSIGNING <FS\_BPHI>.  
      CALL FUNCTION 'NUMERIC\_CHECK'  
        EXPORTING  
          STRING\_IN = <FS\_BPHI>-VERSN  
        IMPORTING  
          HTYPE     = LV\_DTYP.  
  
      IF LV\_DTYP = C\_NUMC.  
        RVAL = <FS\_BPHI>-VERSN.  
      ENDIF.  
      CLEAR LV\_DTYP.  
    ENDLOOP.  
  ENDMETHOD.  
  METHOD SORT\_TABLE.  
    SORT ME->R\_DB\_IO->T\_PSTAT BY OBJNR STAT.  
    SORT ME->R\_DB\_IO->T\_TSTAT BY OBJNR STAT.  
    SORT ME->R\_DB\_IO->T\_RPSCO BY OBJNR WRTTP GJAHR VORGA VERSN BELTP WLP00.  
    SORT ME->R\_DB\_IO->T\_TPART BY SPRAS PARVW.  
    SORT ME->R\_DB\_IO->T\_VBAP BY PS\_PSP\_PNR.  
    SORT ME->R\_DB\_IO->T\_VBPA BY VBELN.  
    SORT ME->R\_DB\_IO->T\_VBPA BY VBELN.  
    SORT ME->R\_DB\_IO->T\_TJ02T BY ISTAT SPRAS.  
    SORT ME->R\_DB\_IO->T\_COEP BY OBJNR.  
    SORT ME->R\_DB\_IO->T\_BPGE BY OBJNR WRTTP.  
    SORT ME->R\_DB\_IO->T\_COOI BY OBJNR.  
    SORT ME->R\_DB\_IO->T\_TCJ1T BY PRART.  
    SORT ME->R\_DB\_IO->T\_BPHI BY VERSN DESCENDING.  
  ENDMETHOD.  
  METHOD DB\_CALL.  
    CREATE OBJECT R\_PERSIST\_DB.  
    R\_PERSIST\_DB->FETCH\_DATA( CHANGING RC\_DATA = R\_DB\_IO ).  
  ENDMETHOD.  
  METHOD TBL\_CONT\_MSGS.  
    IF R\_DB\_IO->T\_PRPS IS INITIAL.  
      LCL\_LOG\_UTILITY=>POPULATE\_LOG( EXPORTING  
                                       I\_TYPE   = C\_E  
                                       I\_CL     = C\_MESS\_CL  
                                       I\_NUMBER = 013  
                                     CHANGING  
                                       C\_LOG    = T\_LOG ).  
    ENDIF.  
    IF R\_DB\_IO->T\_PROJ IS INITIAL.  
      LCL\_LOG\_UTILITY=>POPULATE\_LOG( EXPORTING  
                                       I\_TYPE   = C\_E  
                                       I\_CL     = C\_MESS\_CL  
                                       I\_NUMBER = 014  
                                     CHANGING  
                                       C\_LOG    = T\_LOG ).  
    ENDIF.  
  ENDMETHOD.  
  METHOD FORMAT\_DATA.  
  
    DATA:  
      LV\_KEEP\_RECORD   TYPE BOOLEAN,  
      LV\_POSID         TYPE PS\_POSID,  
      LV\_STATUS        TYPE J\_STATUS,  
      LV\_ORD\_COST      TYPE BP\_WPL,  
      LS\_LVL\_ONE       TYPE \_OUTPUT,  
      LS\_LVL\_ZERO      TYPE \_OUTPUT,  
      LV\_VERSION       TYPE BP\_VERSION.  
  
    FIELD-SYMBOLS:  
      <FS\_PRPS>    LIKE LINE OF R\_DB\_IO->T\_PRPS,  
      <FS\_RPSCO>     TYPE \_RPSCO,  
      <FS\_OUTPUT>  LIKE LINE OF T\_OUTPUT,  
      <FS\_OPTREE>  LIKE LINE OF T\_OUTPUT,  
      <FS\_TCJ1T>   LIKE LINE OF R\_DB\_IO->T\_TCJ1T,  
      <FS\_PROJ>    LIKE LINE OF R\_DB\_IO->T\_PROJ.  
  
    LOOP AT ME->R\_DB\_IO->T\_PROJ ASSIGNING <FS\_PROJ>.  
      CLEAR:  
        LV\_KEEP\_RECORD,  
        LS\_LVL\_ONE.  
      APPEND INITIAL LINE TO T\_OPTREE ASSIGNING <FS\_OPTREE>.  
      CALL FUNCTION 'CONVERSION\_EXIT\_PROJN\_OUTPUT'  
        EXPORTING  
          INPUT  = <FS\_PROJ>-PSPID  
        IMPORTING  
          OUTPUT = <FS\_OPTREE>-PROJ\_PSPID.  
      <FS\_OPTREE>-PROJ\_POST1  = <FS\_PROJ>-POST1.  
      <FS\_OPTREE>-OBJNR = <FS\_PROJ>-OBJNR.  
      <FS\_OPTREE>-PSPNR = <FS\_PROJ>-PSPNR.  
      <FS\_OPTREE>-COMP\_NAME = ME->R\_DB\_IO->COMP\_NAME.  
      <FS\_OPTREE>-PROJ\_PLSEZ  = <FS\_PROJ>-PLSEZ.  
      <FS\_OPTREE>-LEVEL = 1.  
      REPLACE ALL OCCURRENCES OF C\_COMMA  
            IN <FS\_OPTREE>-PROJ\_POST1  
            WITH ABAP\_FALSE.  
      <FS\_OPTREE>-STEP\_NAME = <FS\_OPTREE>-PROJ\_POST1 .  
      LS\_LVL\_ZERO =    <FS\_OPTREE>.  
      LOOP AT ME->R\_DB\_IO->T\_PRPS ASSIGNING <FS\_PRPS>  
           WHERE PSPHI = <FS\_PROJ>-PSPNR.  
        LV\_KEEP\_RECORD = FILTER\_WBS\_ELEMS( I\_STR1 = <FS\_PRPS>  
                                          I\_STR2 = <FS\_PROJ> ).  
        IF LV\_KEEP\_RECORD = ABAP\_TRUE.  
\*\*\*Keep record...  
\*\*\*Use this as a holding place for level one for summation  
          IF <FS\_PRPS>-STUFE = C\_1.  
            LS\_LVL\_ONE-POSID = <FS\_PRPS>-POSID.  
          ENDIF.  
          APPEND INITIAL LINE TO T\_OPTREE ASSIGNING <FS\_OPTREE>.  
          APPEND INITIAL LINE TO T\_OUTPUT ASSIGNING <FS\_OUTPUT>.  
          READ TABLE ME->R\_DB\_IO->T\_TCJ1T  
            ASSIGNING <FS\_TCJ1T>  
            WITH KEY PRART = <FS\_PRPS>-PRART  
            BINARY SEARCH.  
          IF SY-SUBRC = 0.  
            <FS\_OUTPUT>-PROJ\_TYPE  = <FS\_TCJ1T>-PRATX.  
          ENDIF.  
          <FS\_OUTPUT>-COMP\_NAME  = ME->R\_DB\_IO->COMP\_NAME.  
          CLEAR LV\_STATUS.  
          LV\_STATUS = GET\_STATTXT( I\_TASK = <FS\_PRPS>-OBJNR ).  
          <FS\_OUTPUT>-TASK\_STAT = LV\_STATUS(1).  
          CLEAR LV\_STATUS.  
          LV\_STATUS = GET\_STATTXT( I\_PROJ = <FS\_PROJ>-OBJNR ).  
          <FS\_OUTPUT>-PROJ\_STAT = LV\_STATUS(1).  
          <FS\_OUTPUT>-TASK\_DATE   = <FS\_PRPS>-ERDAT.  
          <FS\_OUTPUT>-PROJ\_PSPID  = <FS\_PROJ>-PSPID.  
          <FS\_OUTPUT>-PROJ\_POST1  = <FS\_PROJ>-POST1.  
\*\*\*Replace comma with nothing. ABAP\_FALSE is ''.  
          REPLACE ALL OCCURRENCES OF C\_COMMA  
            IN <FS\_OUTPUT>-PROJ\_POST1  
            WITH ABAP\_FALSE.  
          <FS\_OUTPUT>-PROJ\_PLSEZ  = <FS\_PROJ>-PLSEZ.  
\*\*\*Sales Price  
          READ TABLE ME->R\_DB\_IO->T\_RPSCO  
            ASSIGNING <FS\_RPSCO>  
            WITH KEY OBJNR = <FS\_PRPS>-OBJNR  
                    WRTTP = C\_01  
                    GJAHR = C\_0000  
                    VORGA = C\_KSTR  
                    VERSN = C\_ZERO  
                    BELTP = C\_2  
            BINARY SEARCH.  
          IF SY-SUBRC = 0.  
            LS\_LVL\_ONE-SLS\_PRICE = <FS\_OUTPUT>-SLS\_PRICE = <FS\_RPSCO>-WLP00.  
          ENDIF.  
          <FS\_OUTPUT>-PRPS\_USR00  = <FS\_PRPS>-USR00.  
          <FS\_OUTPUT>-PRPS\_POSKI  = <FS\_PRPS>-POSKI.  
          <FS\_OUTPUT>-PRPS\_POST1  = <FS\_PRPS>-POST1.  
          <FS\_OUTPUT>-DEPT  = <FS\_PRPS>-USR00+0(1).  
\*\*\*Original Cost  
          READ TABLE ME->R\_DB\_IO->T\_RPSCO  
            ASSIGNING <FS\_RPSCO>  
            WITH KEY OBJNR = <FS\_PRPS>-OBJNR  
                    WRTTP = C\_01  
                    VORGA = C\_KSTP  
                    VERSN = C\_001  
                    BELTP = C\_1  
            BINARY SEARCH.  
          IF SY-SUBRC = 0.  
            IF <FS\_PRPS>-POSID = LS\_LVL\_ONE-POSID.  
              "THIS SHOULD BE CLEAR FOR LEVEL ONE, BUT JUST INCASE IT IS NOT  
              CLEAR <FS\_RPSCO>-WLP00.  
            ENDIF.  
            <FS\_OUTPUT>-ORIG\_COST = <FS\_RPSCO>-WLP00.  
            LS\_LVL\_ONE-ORIG\_COST  = LS\_LVL\_ONE-ORIG\_COST + <FS\_OUTPUT>-ORIG\_COST.  
          ENDIF.  
  
\*\*\*Current Budget  
          LV\_VERSION = GET\_VERSION( <FS\_PROJ> ).  
          UNPACK LV\_VERSION TO LV\_VERSION.  
          READ TABLE ME->R\_DB\_IO->T\_RPSCO  
            ASSIGNING <FS\_RPSCO>  
            WITH KEY OBJNR = <FS\_PRPS>-OBJNR  
                    WRTTP = C\_01  
                    VORGA = C\_KSTP  
                    VERSN = LV\_VERSION  
                    BELTP = C\_1  
            BINARY SEARCH.  
          IF SY-SUBRC = 0.  
            IF <FS\_PRPS>-POSID = LS\_LVL\_ONE-POSID.  
              "THIS SHOULD BE CLEAR FOR LEVEL ONE, BUT JUST INCASE IT IS NOT  
              CLEAR <FS\_RPSCO>-WLP00.  
            ENDIF.  
            <FS\_OUTPUT>-CURR\_BUDG  = <FS\_RPSCO>-WLP00.  
            LS\_LVL\_ONE-CURR\_BUDG  = LS\_LVL\_ONE-CURR\_BUDG + <FS\_OUTPUT>-CURR\_BUDG.  
          ENDIF.  
  
\*\*\*Projected End Cost  
          READ TABLE ME->R\_DB\_IO->T\_RPSCO  
            ASSIGNING <FS\_RPSCO>  
            WITH KEY OBJNR = <FS\_PRPS>-OBJNR  
                    WRTTP = C\_01  
                    VORGA = C\_KSTP  
                    VERSN = C\_ZERO  
                    BELTP = C\_1  
            BINARY SEARCH.  
          IF SY-SUBRC = 0.  
            IF <FS\_PRPS>-POSID = LS\_LVL\_ONE-POSID.  
              "THIS SHOULD BE CLEAR FOR LEVEL ONE, BUT JUST INCASE IT IS NOT  
              CLEAR <FS\_RPSCO>-WLP00.  
            ENDIF.  
            <FS\_OUTPUT>-END\_COST  = <FS\_RPSCO>-WLP00.  
            LS\_LVL\_ONE-END\_COST  = LS\_LVL\_ONE-END\_COST + <FS\_OUTPUT>-END\_COST.  
          ENDIF.  
          IF <FS\_PRPS>-STUFE = C\_1.  
            "LEVEL ONE  
          ELSE.  
            GET\_ACTUALS( EXPORTING I\_VAR  = <FS\_PRPS>-OBJNR  
                         CHANGING  I\_STR1 = <FS\_OUTPUT> ).  
            LS\_LVL\_ONE-ACTL\_COST  = LS\_LVL\_ONE-ACTL\_COST + <FS\_OUTPUT>-ACTL\_COST.  
  
            GET\_ORDERED( EXPORTING I\_VAR  = <FS\_PRPS>-OBJNR  
                         CHANGING  I\_STR1 = <FS\_OUTPUT> ).  
            LS\_LVL\_ONE-ORDR\_COST    = LS\_LVL\_ONE-ORDR\_COST + <FS\_OUTPUT>-ORDR\_COST.  
          ENDIF.  
  
\*\*\*Ordered Percent  
          CLEAR LV\_ORD\_COST.  
          IF <FS\_OUTPUT>-END\_COST IS NOT INITIAL.  
            LV\_ORD\_COST = ( <FS\_OUTPUT>-ORDR\_COST / <FS\_OUTPUT>-END\_COST ) \* 100.  
            WRITE LV\_ORD\_COST TO <FS\_OUTPUT>-ORDR\_PERC DECIMALS 2 RIGHT-JUSTIFIED.  
            CONCATENATE  <FS\_OUTPUT>-ORDR\_PERC C\_% INTO <FS\_OUTPUT>-ORDR\_PERC.  
          ELSE.  
            WRITE LV\_ORD\_COST TO <FS\_OUTPUT>-ORDR\_PERC DECIMALS 2 RIGHT-JUSTIFIED.  
            CONCATENATE  <FS\_OUTPUT>-ORDR\_PERC C\_% INTO <FS\_OUTPUT>-ORDR\_PERC.  
          ENDIF.  
  
\*\*\*Cur Budget Less Actual Cost  
          <FS\_OUTPUT>-CURR\_ACTL = <FS\_OUTPUT>-ORIG\_COST - <FS\_OUTPUT>-ORDR\_COST.  
  
          <FS\_OUTPUT>-VBUKR = P\_VBUKR.  
          <FS\_OUTPUT>-VKOKR = P\_VKOKR.  
          <FS\_OUTPUT>-PSPNR = <FS\_PROJ>-PSPNR.  
          <FS\_OUTPUT>-POSID = <FS\_PRPS>-POSID.  
          <FS\_OUTPUT>-PRCTR = <FS\_PRPS>-PRCTR.  
          <FS\_OUTPUT>-PSPHI = <FS\_PRPS>-PSPHI.  
          <FS\_OUTPUT>-OBJNR = <FS\_PROJ>-OBJNR.  
          CONV\_ELEMENTS( CHANGING STR1 = <FS\_OUTPUT> ).  
          GET\_CUSTDATA(  EXPORTING  
                               I\_STR1 = <FS\_PRPS>  
                         IMPORTING  
                               E\_PARVW = <FS\_OUTPUT>-PARVW  
                               E\_KUNNR = <FS\_OUTPUT>-KUNNR ).  
          <FS\_OPTREE> = <FS\_OUTPUT>.  
          <FS\_OPTREE>-STEP\_NAME = <FS\_OUTPUT>-PRPS\_POST1 .  
          CLEAR <FS\_OPTREE>-PROJ\_POST1.  
          CLEAR <FS\_OPTREE>-PROJ\_PLSEZ.  
          IF <FS\_PRPS>-STUFE = C\_1.  
            <FS\_OPTREE>-LEVEL = 2.  
          ELSE.  
            <FS\_OPTREE>-LEVEL = 3.  
          ENDIF.  
          CLEAR: LV\_KEEP\_RECORD.  
          IF <FS\_PRPS>-STUFE <> C\_1.  
            LV\_KEEP\_RECORD = CHECK\_VALUES( <FS\_OUTPUT> ).  
            IF LV\_KEEP\_RECORD = ABAP\_FALSE.  
              LV\_POSID = <FS\_OUTPUT>-POSID.  
              UNASSIGN <FS\_OUTPUT>.  
              READ TABLE T\_OUTPUT ASSIGNING <FS\_OUTPUT>  
                WITH KEY POSID = LV\_POSID.  
              IF SY-SUBRC = 0.  
                DELETE T\_OUTPUT INDEX SY-TABIX.  
                DELETE T\_OPTREE INDEX SY-TABIX.  
                CONTINUE.  
              ENDIF.  
            ENDIF.  
          ENDIF.  
        ELSE.  
          CONTINUE.  
        ENDIF.  
      ENDLOOP.  
  
      CALL FUNCTION 'CONVERSION\_EXIT\_PROJN\_OUTPUT'  
        EXPORTING  
          INPUT  = LS\_LVL\_ONE-POSID  
        IMPORTING  
          OUTPUT = LS\_LVL\_ONE-POSID.  
      SORT T\_OUTPUT BY POSID.  
      READ TABLE T\_OUTPUT  
        ASSIGNING <FS\_OUTPUT>  
        WITH KEY POSID = LS\_LVL\_ONE-POSID.  
      IF SY-SUBRC = 0.  
        <FS\_OUTPUT>-ORIG\_COST = LS\_LVL\_ONE-ORIG\_COST.  
        <FS\_OUTPUT>-CURR\_BUDG = LS\_LVL\_ONE-CURR\_BUDG.  
        <FS\_OUTPUT>-END\_COST  = LS\_LVL\_ONE-END\_COST.  
        <FS\_OUTPUT>-ACTL\_COST = LS\_LVL\_ONE-ACTL\_COST.  
        <FS\_OUTPUT>-ORDR\_COST = LS\_LVL\_ONE-ORDR\_COST.  
  
        CLEAR LV\_ORD\_COST.  
        IF LS\_LVL\_ONE-END\_COST IS NOT INITIAL.  
          LV\_ORD\_COST = ( LS\_LVL\_ONE-ORDR\_COST / LS\_LVL\_ONE-END\_COST ) \* 100.  
          WRITE LV\_ORD\_COST TO <FS\_OUTPUT>-ORDR\_PERC DECIMALS 2 RIGHT-JUSTIFIED.  
          CONCATENATE  <FS\_OUTPUT>-ORDR\_PERC C\_% INTO <FS\_OUTPUT>-ORDR\_PERC.  
        ELSE.  
          WRITE LV\_ORD\_COST TO <FS\_OUTPUT>-ORDR\_PERC DECIMALS 2 RIGHT-JUSTIFIED.  
          CONCATENATE  <FS\_OUTPUT>-ORDR\_PERC C\_% INTO <FS\_OUTPUT>-ORDR\_PERC.  
        ENDIF.  
\*        <FS\_OUTPUT>-ORDR\_PERC = LS\_LVL\_ONE-ORDR\_COST / LS\_LVL\_ONE-END\_COST.  
        <FS\_OUTPUT>-CURR\_ACTL = LS\_LVL\_ONE-ORIG\_COST - LS\_LVL\_ONE-ORDR\_COST.  
      ENDIF.  
      SORT T\_OPTREE BY PROJ\_PSPID PSPNR POSID.  
      READ TABLE T\_OPTREE  
        ASSIGNING <FS\_OPTREE>  
        WITH KEY COMP\_NAME = LS\_LVL\_ZERO-COMP\_NAME  
                 PROJ\_PSPID = LS\_LVL\_ZERO-PROJ\_PSPID  
                 PROJ\_POST1 = LS\_LVL\_ZERO-PROJ\_POST1  
                 POSID = SPACE.  
      IF SY-SUBRC = 0.  
        IF <FS\_OUTPUT> IS ASSIGNED.  
          <FS\_OPTREE>-ORIG\_COST = <FS\_OUTPUT>-ORIG\_COST.  
          <FS\_OPTREE>-CURR\_BUDG = <FS\_OUTPUT>-CURR\_BUDG.  
          <FS\_OPTREE>-END\_COST  = <FS\_OUTPUT>-END\_COST.  
          <FS\_OPTREE>-ACTL\_COST = <FS\_OUTPUT>-ACTL\_COST.  
          <FS\_OPTREE>-ORDR\_COST = <FS\_OUTPUT>-ORDR\_COST.  
          <FS\_OPTREE>-ORDR\_PERC = <FS\_OUTPUT>-ORDR\_PERC.  
          <FS\_OPTREE>-CURR\_ACTL = <FS\_OUTPUT>-CURR\_ACTL.  
          <FS\_OPTREE>-PROJ\_STAT = LV\_STATUS(1).  
          <FS\_OPTREE>-SLS\_PRICE = LS\_LVL\_ONE-SLS\_PRICE .  
        ENDIF.  
      ENDIF.  
      READ TABLE T\_OPTREE  
         ASSIGNING <FS\_OPTREE>  
         WITH KEY POSID = LS\_LVL\_ONE-POSID.  
      IF SY-SUBRC = 0.  
        IF <FS\_OUTPUT> IS ASSIGNED.  
          <FS\_OPTREE>-ORIG\_COST = <FS\_OUTPUT>-ORIG\_COST.  
          <FS\_OPTREE>-CURR\_BUDG = <FS\_OUTPUT>-CURR\_BUDG.  
          <FS\_OPTREE>-END\_COST  = <FS\_OUTPUT>-END\_COST.  
          <FS\_OPTREE>-ACTL\_COST = <FS\_OUTPUT>-ACTL\_COST.  
          <FS\_OPTREE>-ORDR\_COST = <FS\_OUTPUT>-ORDR\_COST.  
          <FS\_OPTREE>-ORDR\_PERC = <FS\_OUTPUT>-ORDR\_PERC.  
          <FS\_OPTREE>-CURR\_ACTL = <FS\_OUTPUT>-CURR\_ACTL.  
          <FS\_OPTREE>-PROJ\_STAT = LV\_STATUS(1).  
        ENDIF.  
      ENDIF.  
    ENDLOOP.  
    DESCRIBE TABLE T\_OUTPUT LINES V\_RECORDS.  
  ENDMETHOD.  
  METHOD CHECK\_VALUES.  
    DATA:  
      S\_STR  TYPE \_OUTPUT.  
    S\_STR = I\_STR.  
    IF S\_STR-ORIG\_COST IS INITIAL AND  
       S\_STR-CURR\_BUDG IS INITIAL AND  
       S\_STR-END\_COST  IS INITIAL AND  
       S\_STR-ACTL\_COST IS INITIAL AND  
       S\_STR-ORDR\_COST IS INITIAL AND  
       S\_STR-SLS\_PRICE IS INITIAL.  
      RVAL = ABAP\_FALSE.  
    ELSE.  
      RVAL = ABAP\_TRUE.  
    ENDIF.  
  ENDMETHOD.  
ENDCLASS.  
CLASS LCL\_CNTRL DEFINITION.  
  PUBLIC SECTION.  
    METHODS: CONSTRUCTOR,  
                   START.  
    DATA: R\_CL\_MODEL TYPE REF TO LCL\_MODEL,  
          R\_CL\_VIEW  TYPE REF TO LCL\_VIEW.  
  
ENDCLASS.  
CLASS LCL\_CNTRL IMPLEMENTATION.  
  METHOD CONSTRUCTOR.  
    CREATE OBJECT: R\_CL\_MODEL,R\_CL\_VIEW.  
  ENDMETHOD.  
  METHOD START.  
    R\_CL\_VIEW->GET\_PARAMS( IMPORTING  
        VBUKR = R\_CL\_MODEL->R\_DB\_IO->VBUKR  
        VKOKR = R\_CL\_MODEL->R\_DB\_IO->VKOKR  
        STAT = R\_CL\_MODEL->R\_DB\_IO->STAT  
        EXPATH = R\_CL\_MODEL->R\_DB\_IO->EXPATH  
        PSPID = R\_CL\_MODEL->R\_DB\_IO->PSPID[]  
        POSID = R\_CL\_MODEL->R\_DB\_IO->POSID[]  
        POSKI = R\_CL\_MODEL->R\_DB\_IO->POSKI[]  
        PRCTR = R\_CL\_MODEL->R\_DB\_IO->PRCTR[]  
        PRART = R\_CL\_MODEL->R\_DB\_IO->PRART[] ).  
    CHECK R\_CL\_MODEL->ACCESS\_DATA( ) <> ABAP\_TRUE.  
    R\_CL\_MODEL->FORMAT\_DATA( ).  
    IF SY-BATCH IS INITIAL.  
      R\_CL\_VIEW->DISP\_TREE( EXPORTING T\_OP = R\_CL\_MODEL->T\_OPTREE  
                              COMP\_NAME = R\_CL\_MODEL->R\_DB\_IO->COMP\_NAME ).  
    ELSE.  
      R\_CL\_VIEW->SAVE\_FILE( IMPORTING T\_LOG = R\_CL\_MODEL->T\_LOG  ).  
    ENDIF.  
  
  ENDMETHOD.  
ENDCLASS.  
  
  
DATA LCL\_MAIN TYPE REF TO LCL\_CNTRL.  
  
INITIALIZATION.  
  CREATE OBJECT: LCL\_MAIN.  
  
  
AT SELECTION-SCREEN.  
  IF SY-BATCH IS NOT INITIAL.  
    LCL\_MAIN->R\_CL\_VIEW->CHECK\_FILE( ).  
  ENDIF.  
  
AT SELECTION-SCREEN ON VALUE-REQUEST FOR P\_EXPATH.  
  LCL\_MAIN->R\_CL\_VIEW->F4\_HLPFILE( CHANGING EXPATH = P\_EXPATH ).  
  
START-OF-SELECTION.  
  
  LCL\_MAIN->START( ).  
  
END-OF-SELECTION.  
  CHECK SY-BATCH IS NOT INITIAL.  
  LCL\_MAIN->R\_CL\_VIEW->WRITE\_LOG( EXPORTING EXPATH = P\_EXPATH ).