\*&---------------------------------------------------------------------\*  
\*& Report  ZDAC\_PRJCSR  
\*&  
\*&---------------------------------------------------------------------\*  
\*&  
\*&  
\*&---------------------------------------------------------------------\*  
REPORT ZDAC\_PROJ\_COST\_SUMM NO STANDARD PAGE HEADING.  
TABLES: PROJ,PRPS,TCNT.  
\*-----------------------------------------------------------------------  
\* 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 \_BPHI,  
    OBJNR  TYPE   BP\_OBJHIER,  
    WRTTP  TYPE   CO\_WRTTP,  
    HIERA  TYPE   BP\_HIERID,  
    GEBER  TYPE   BP\_GEBER,  
    VERSN  TYPE   BP\_VERSION,  
    FAREA  TYPE   FM\_FAREA,  
  END OF \_BPHI,  
  BEGIN OF \_COOI,  
    RFPOS  TYPE   CC\_RFPOS,  
    RFKNT  TYPE   CC\_RFKNT,  
    RFTRM  TYPE   CC\_RFTRM,  
    RFART  TYPE   CC\_RFART,  
    LIFNR  TYPE   LIFNR,  
    LEDNR  TYPE   LEDNR,  
    OBJNR  TYPE   J\_OBJNR,  
    HRKFT  TYPE   CO\_SUBKEY,  
    RFORG  TYPE   AWORG,  
    RFTYP  TYPE   AWTYP,  
    RFSYS  TYPE   AWSYS,  
    VRGNG  TYPE   CO\_VORGANG,  
    GJAHR  TYPE   GJAHR,  
    WRTTP  TYPE   CO\_WRTTP,  
    PERIO  TYPE   CO\_PERIO,  
    BUKRS  TYPE   BUKRS,  
    KOKRS  TYPE   KOKRS,  
    WHGBTR TYPE WHGXXX,  
    WOGBTR TYPE WOGXXX,  
  END OF \_COOI,  
  BEGIN OF \_COEP,  
    KOKRS  TYPE   KOKRS,  
    BELNR  TYPE   CO\_BELNR,  
    BUZEI  TYPE   CO\_BUZEI,  
    PERIO  TYPE   CO\_PERIO,  
    LEDNR  TYPE   LEDNR,  
    OBJNR  TYPE   J\_OBJNR,  
    GJAHR  TYPE   GJAHR,  
    WRTTP  TYPE   CO\_WRTTP,  
    VERSN  TYPE   VERSN,  
    KSTAR  TYPE   KSTAR,  
    WTGBTR  TYPE  WTGXXX,  
  END OF \_COEP,  
  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 \_VBPA,  
    VBELN  TYPE   VBELN,  
    POSNR  TYPE   POSNR,  
    PARVW  TYPE   PARVW,  
    KUNNR  TYPE   KUNNR,  
  END OF \_VBPA,  
  BEGIN OF \_VBAP,  
    VBELN  TYPE   VBELN,  
    POSNR  TYPE   POSNR,  
    MATNR  TYPE   MATNR,  
    PS\_PSP\_PNR  TYPE  PS\_PSP\_PNR,  
  END OF \_VBAP,  
  BEGIN OF \_KNA1,  
    KUNNR  TYPE   KUNNR,  
    NAME1  TYPE   NAME1,  
    NAME2  TYPE   NAME2,  
  END OF \_KNA1,  
  BEGIN OF \_OUTPUT,  
    COMP\_NAME  TYPE NAME\_1,  
    STEP\_NAME(30),  
    DEPT(1),  
    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,  
    SLS\_PRICE\_C(20) TYPE C,  
    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,  
    LEVEL TYPE I,  
    SUBTOT,  
  END OF \_OUTPUT,  
  BEGIN  OF   \_DEPTOT,  
    DEPT(1),  
    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,  
  END OF \_DEPTOT.  
CONSTANTS:C\_MESS\_CL   TYPE SY-MSGID        VALUE 'ZPSDVRP',  
          C\_E         TYPE BAPIRETURN-TYPE VALUE 'E',  
          C\_EQ        TYPE CHAR2           VALUE 'EQ',  
          C\_BT        TYPE CHAR2           VALUE 'BT',  
          C\_A         TYPE BAPIRETURN-TYPE VALUE 'A',  
          C\_F         TYPE C               VALUE 'F',  
          C\_I         TYPE BAPIRETURN-TYPE VALUE 'I',  
          C\_J         TYPE C               VALUE 'J',  
          C\_M         TYPE C               VALUE 'M',  
          C\_S         TYPE C               VALUE 'S',  
          C\_X         TYPE C               VALUE 'X',  
          C\_W         TYPE BAPIRETURN-TYPE VALUE 'W',  
          C\_CSV(5)                         VALUE '.CSV',  
          C\_CSV2(3)                        VALUE 'CSV',  
          C\_SEP       TYPE C               VALUE ',',  
          C\_QUOTE       TYPE C             VALUE '"',  
          C\_PERIOD    TYPE C               VALUE '.',  
          C\_COMMA     TYPE C               VALUE ',',  
          C\_0         TYPE C               VALUE '0',  
          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\_22(2)     TYPE C               VALUE '22',  
          C\_24(2)     TYPE C               VALUE '24',  
          C\_26(2)     TYPE C               VALUE '26',  
          C\_02(2)     TYPE C               VALUE '02',  
          C\_04(2)     TYPE C               VALUE '04',  
          C\_11(2)     TYPE C               VALUE '11',  
          C\_42(2)     TYPE C               VALUE '42',  
          C\_2         TYPE C               VALUE '2',  
          C\_001(3)                         VALUE '001',  
          C\_002(3)                         VALUE '002',  
          C\_I0002(5)                       VALUE 'I0002',  
          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\_2999(4)   TYPE C               VALUE '2999',  
          C\_NUMC(4)                        VALUE 'NUMC',  
          C\_KSTR(4)   TYPE C               VALUE 'KSTR',  
          C\_KSTP(4)   TYPE C               VALUE 'KSTP'.  
\*----------------------------------------------------------------------\*  
\*       CLASS LCL\_DB\_IO DEFINITION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
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,  
          PROF\_DB TYPE PS\_PROF\_DB,  
          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.                    "LCL\_DB\_IO DEFINITION  
CLASS LCL\_DB\_IO IMPLEMENTATION.  
  METHOD CONSTRUCTOR.  
    CLEAR: ME->T\_PROJ[],  
       ME->T\_PRPS[],  
       ME->T\_PSTAT[],  
       ME->T\_TSTAT[],  
       ME->T\_TPART[],  
       ME->T\_KNA1[],  
       ME->T\_VBAP[],  
       ME->T\_VBPA[],  
       ME->T\_RPSCO[],  
       ME->T\_TJ02T[],  
       ME->T\_COEP[],  
\*       ME->T\_BPGE[],  
       ME->T\_BPHI[],  
       ME->T\_TCJ1T[],  
       ME->T\_COOI[],  
       ME->COMP\_NAME.  
  ENDMETHOD.  
ENDCLASS.  
\*----------------------------------------------------------------------\*  
\*       CLASS LCL\_PERSIST DEFINITION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
CLASS LCL\_PERSIST DEFINITION.  
  PUBLIC SECTION.  
    METHODS:  
     FETCH\_DATA CHANGING R\_DBIO TYPE REF TO LCL\_DB\_IO.  
ENDCLASS.                    "LCL\_PERSIST DEFINITION  
  
\*----------------------------------------------------------------------\*  
\*       CLASS LCL\_PERSIST IMPLEMENTATION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
CLASS LCL\_PERSIST IMPLEMENTATION.  
  METHOD FETCH\_DATA.  
  
    DATA:  
      LV\_COMPCODE    TYPE RCOMP\_D.  
  
    UNPACK R\_DBIO->VBUKR TO LV\_COMPCODE.  
    SELECT SINGLE NAME1 INTO R\_DBIO->COMP\_NAME FROM T880  
      WHERE RCOMP = LV\_COMPCODE.  
  
    IF R\_DBIO->PSPID IS NOT INITIAL.  
      SELECT VBUKR VKOKR PSPNR PSPID POST1 PLSEZ OBJNR FROM PROJ  
        INTO CORRESPONDING FIELDS OF TABLE R\_DBIO->T\_PROJ  
        WHERE PSPID IN R\_DBIO->PSPID AND  
              VBUKR  = R\_DBIO->VBUKR AND  
              VKOKR  = R\_DBIO->VKOKR.  
      IF R\_DBIO->T\_PROJ IS NOT INITIAL.  
        SELECT POSID POSKI PRCTR PRART POST1  
               PSPHI OBJNR USR00 PSPNR ERDAT  
               STUFE  
          FROM PRPS  
          INTO CORRESPONDING FIELDS OF TABLE R\_DBIO->T\_PRPS  
          FOR ALL ENTRIES IN R\_DBIO->T\_PROJ  
          WHERE PSPHI  = R\_DBIO->T\_PROJ-PSPNR AND  
                PRCTR IN R\_DBIO->PRCTR        AND  
                PRART IN R\_DBIO->PRART        AND  
                POSKI IN R\_DBIO->POSKI.  
      ENDIF.  
    ELSE.  
      IF R\_DBIO->POSID[] IS NOT INITIAL.  
        SELECT POSID POSKI PRCTR PRART POST1  
               PSPHI OBJNR USR00 PSPNR ERDAT  
               STUFE  
          FROM PRPS  
          INTO CORRESPONDING FIELDS OF TABLE R\_DBIO->T\_PRPS  
          WHERE POSID IN R\_DBIO->POSID  AND  
                PRCTR IN R\_DBIO->PRCTR  AND  
                PRART IN R\_DBIO->PRART  AND  
                POSKI IN R\_DBIO->POSKI.  
        IF R\_DBIO->T\_PRPS IS NOT INITIAL.  
          SELECT VBUKR VKOKR PSPNR PSPID POST1  
                 PLSEZ OBJNR  
            FROM PROJ  
            INTO CORRESPONDING FIELDS OF TABLE R\_DBIO->T\_PROJ  
            FOR ALL ENTRIES IN R\_DBIO->T\_PRPS  
            WHERE PSPNR  = R\_DBIO->T\_PRPS-PSPHI AND  
                  VBUKR  = R\_DBIO->VBUKR      AND  
                  VKOKR  = R\_DBIO->VKOKR.  
        ENDIF.  
        IF R\_DBIO->T\_PRPS IS NOT INITIAL.  
          SELECT POSID POSKI PRCTR PRART POST1  
                 PSPHI OBJNR USR00 PSPNR ERDAT  
                 STUFE  
            FROM PRPS  
            INTO CORRESPONDING FIELDS OF TABLE R\_DBIO->T\_PRPS  
            FOR ALL ENTRIES IN R\_DBIO->T\_PROJ  
            WHERE PSPHI  = R\_DBIO->T\_PROJ-PSPNR AND  
                  PRCTR IN R\_DBIO->PRCTR      AND  
                  PRART IN R\_DBIO->PRART      AND  
                  POSKI IN R\_DBIO->POSKI.  
        ENDIF.  
      ELSE.  
        IF R\_DBIO->POSKI IS NOT INITIAL.  
  
          SELECT POSID POSKI PRCTR PRART POST1  
                 PSPHI OBJNR USR00 PSPNR ERDAT  
                 STUFE  
            FROM PRPS  
            INTO CORRESPONDING FIELDS OF TABLE R\_DBIO->T\_PRPS  
            WHERE POSKI IN R\_DBIO->POSKI  AND  
                  PRCTR IN R\_DBIO->PRCTR  AND  
                  PRART IN R\_DBIO->PRART.  
          IF R\_DBIO->T\_PRPS IS NOT INITIAL.  
            SELECT VBUKR VKOKR PSPNR PSPID POST1  
                   PLSEZ OBJNR  
              FROM PROJ  
              INTO CORRESPONDING FIELDS OF TABLE R\_DBIO->T\_PROJ  
              FOR ALL ENTRIES IN R\_DBIO->T\_PRPS  
              WHERE PSPNR  = R\_DBIO->T\_PRPS-PSPHI AND  
                    VBUKR  = R\_DBIO->VBUKR        AND  
                    VKOKR  = R\_DBIO->VKOKR.  
          ENDIF.  
          IF R\_DBIO->T\_PROJ IS NOT INITIAL.  
            SELECT POSID POSKI PRCTR PRART POST1 PSPHI  
                   OBJNR USR00 PSPNR ERDAT STUFE  
              FROM PRPS  
              INTO CORRESPONDING FIELDS OF TABLE R\_DBIO->T\_PRPS  
              FOR ALL ENTRIES IN R\_DBIO->T\_PROJ  
              WHERE PSPHI  = R\_DBIO->T\_PROJ-PSPNR AND  
                    PRCTR IN R\_DBIO->PRCTR        AND  
                    PRART IN R\_DBIO->PRART        AND  
                    POSKI IN R\_DBIO->POSKI.  
          ENDIF.  
        ENDIF.  
      ENDIF.  
    ENDIF.  
  
    IF R\_DBIO->T\_PROJ IS NOT INITIAL.  
      SELECT \* FROM JEST  
        INTO TABLE R\_DBIO->T\_PSTAT  
        FOR ALL ENTRIES IN R\_DBIO->T\_PROJ  
        WHERE OBJNR = R\_DBIO->T\_PROJ-OBJNR  
        .  
    ENDIF.  
  
    IF R\_DBIO->T\_PRPS IS NOT INITIAL.  
      SELECT \* FROM JEST  
        INTO TABLE R\_DBIO->T\_TSTAT  
        FOR ALL ENTRIES IN R\_DBIO->T\_PRPS  
        WHERE OBJNR = R\_DBIO->T\_PRPS-OBJNR.  
      IF R\_DBIO->T\_TSTAT IS NOT INITIAL.  
        SELECT ISTAT SPRAS TXT04 TXT30  
          FROM TJ02T  
          INTO TABLE R\_DBIO->T\_TJ02T  
          FOR ALL ENTRIES IN R\_DBIO->T\_TSTAT  
          WHERE ISTAT = R\_DBIO->T\_TSTAT-STAT  
          AND SPRAS = SY-LANGU.                         "#EC CI\_GENBUFF  
      ENDIF.  
  
      SELECT OBJNR WRTTP GJAHR VORGA VERSN  
             BELTP WLP00  
        FROM RPSCO  
        INTO TABLE R\_DBIO->T\_RPSCO  
        FOR ALL ENTRIES IN R\_DBIO->T\_PRPS  
        WHERE OBJNR = R\_DBIO->T\_PRPS-OBJNR.  
  
      SELECT VBELN POSNR MATNR PS\_PSP\_PNR  
        FROM VBAP  
        INTO TABLE R\_DBIO->T\_VBAP  
        FOR ALL ENTRIES IN R\_DBIO->T\_PRPS  
        WHERE PS\_PSP\_PNR = R\_DBIO->T\_PRPS-PSPNR. "#EC CI\_NOFIELD #EC CI\_GENBUFF.  
      IF R\_DBIO->T\_VBAP IS NOT INITIAL.  
        SELECT VBELN POSNR PARVW KUNNR  
          FROM VBPA  
          INTO TABLE R\_DBIO->T\_VBPA  
          FOR ALL ENTRIES IN R\_DBIO->T\_VBAP  
          WHERE VBELN = R\_DBIO->T\_VBAP-VBELN.  
      ENDIF.  
      IF R\_DBIO->T\_VBPA IS NOT INITIAL.  
        SELECT KUNNR  NAME1 NAME2  
          FROM KNA1  
          INTO TABLE R\_DBIO->T\_KNA1  
          FOR ALL ENTRIES IN R\_DBIO->T\_VBPA  
          WHERE KUNNR = R\_DBIO->T\_VBPA-KUNNR.  
        IF R\_DBIO->T\_VBPA IS NOT INITIAL.  
          SELECT SPRAS PARVW VTEXT  
                FROM TPART  
                INTO CORRESPONDING FIELDS OF TABLE R\_DBIO->T\_TPART  
                FOR ALL ENTRIES IN R\_DBIO->T\_VBPA  
                WHERE PARVW = R\_DBIO->T\_VBPA-PARVW  
                AND SPRAS = SY-LANGU.  
        ENDIF.  
      ENDIF.  
  
      SELECT KOKRS BELNR BUZEI PERIO LEDNR  
             OBJNR GJAHR WRTTP VERSN KSTAR  
             WTGBTR  
        FROM COEP  
        INTO CORRESPONDING FIELDS OF TABLE R\_DBIO->T\_COEP  
        FOR ALL ENTRIES IN R\_DBIO->T\_PRPS  
        WHERE OBJNR = R\_DBIO->T\_PRPS-OBJNR AND  
              KOKRS = R\_DBIO->VKOKR.  
  
\*      SELECT \*  
\*        FROM BPGE  
\*        INTO TABLE R\_DBIO->T\_BPGE  
\*        FOR ALL ENTRIES IN R\_DBIO->T\_PRPS  
\*        WHERE OBJNR = R\_DBIO->T\_PRPS-OBJNR.  
  
      SELECT RFPOS RFKNT RFTRM RFART  
             LIFNR LEDNR OBJNR HRKFT  
             RFORG RFTYP RFSYS VRGNG  
             GJAHR WRTTP PERIO BUKRS  
             KOKRS WHGBTR WOGBTR  
        FROM COOI  
        INTO CORRESPONDING FIELDS OF TABLE R\_DBIO->T\_COOI  
        FOR ALL ENTRIES IN R\_DBIO->T\_PRPS  
        WHERE OBJNR = R\_DBIO->T\_PRPS-OBJNR.            "#EC CI\_NOFIRST.  
  
      SELECT LANGU PRART PRATX  
        FROM TCJ1T  
        INTO CORRESPONDING FIELDS OF TABLE R\_DBIO->T\_TCJ1T  
        FOR ALL ENTRIES IN R\_DBIO->T\_PRPS  
        WHERE PRART = R\_DBIO->T\_PRPS-PRART.  
  
      SELECT OBJNR  WRTTP HIERA GEBER VERSN FAREA  
        FROM BPHI  
        INTO CORRESPONDING FIELDS OF TABLE R\_DBIO->T\_BPHI  
       FOR ALL ENTRIES IN R\_DBIO->T\_PRPS  
       WHERE OBJNR = R\_DBIO->T\_PRPS-OBJNR AND  
             WRTTP = C\_01.  
    ENDIF.  
  ENDMETHOD.                    "FETCH\_DATA  
ENDCLASS.                    "LCL\_PERSIST IMPLEMENTATION  
\*----------------------------------------------------------------------\*  
\*       CLASS LCL\_CNTLR DEFINITION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
CLASS LCL\_CNTLR DEFINITION.  
  PUBLIC SECTION.  
  
    CLASS-METHODS:  
      DWNLD\_FILE RETURNING VALUE(RVAL) TYPE BOOLEAN,  
      F4\_HLPFILE CHANGING EXPATH TYPE RLGRAP-FILENAME,  
      CHECK\_BFILE IMPORTING BTPATH TYPE RLGRAP-FILENAME  
                  EXPORTING FILE TYPE STRING  
                  RETURNING VALUE(RVAL) TYPE BOOLEAN,  
      CHECK\_FFILE IMPORTING EXPATH TYPE RLGRAP-FILENAME  
                 RETURNING VALUE(RVAL) TYPE BOOLEAN,  
      ERROR\_CHECK RETURNING VALUE(RVAL) TYPE BOOLEAN,  
      WRITE\_LOG IMPORTING EXPATH TYPE RLGRAP-FILENAME,  
      SAVE\_FILE,  
      SET\_BPATH EXPORTING BTPATH TYPE RLGRAP-FILENAME,  
      SCR\_CHECK IMPORTING  
                           VBUKR  TYPE PS\_VBUKR  
                           VKOKR  TYPE PS\_VKOKR  
                           STAT   TYPE J\_STATUS  
                           EXPATH TYPE LOCALFILE  
                           PSPID  TYPE RSELOPTION  
                           POSID  TYPE RSELOPTION  
                           POSKI  TYPE RSELOPTION  
                           PRCTR  TYPE RSELOPTION  
                           PRART  TYPE RSELOPTION  
                RETURNING  VALUE(RVAL) TYPE BOOLEAN.  
    CLASS-DATA: V\_ERROR\_CHECK TYPE BOOLEAN,  
                EXPATH TYPE LOCALFILE,  
                BTPATH TYPE RLGRAP-FILENAME,  
                DWNLFF TYPE BOOLEAN,  
                DWNLBT TYPE BOOLEAN,  
                T\_OUTPUT TYPE TABLE OF \_OUTPUT.  
    METHODS:  
      CONSTRUCTOR IMPORTING  
                           VBUKR  TYPE PS\_VBUKR  
                           VKOKR  TYPE PS\_VKOKR  
                           STAT   TYPE J\_STATUS  
                           PROF\_DB TYPE PS\_PROF\_DB  
                           DWNLFF TYPE BOOLEAN  
                           DWNLBT TYPE BOOLEAN  
                           EXPATH TYPE LOCALFILE  
                           BTPATH TYPE RLGRAP-FILENAME  
                           PSPID  TYPE RSELOPTION  
                           POSID  TYPE RSELOPTION  
                           POSKI  TYPE RSELOPTION  
                           PRCTR  TYPE RSELOPTION  
                           PRART  TYPE RSELOPTION,  
      CREATE\_CONTROLS,  
\*      FREE\_CONTROLS,  
      INIT\_ALV  
\*      PAI           IMPORTING OK\_CODE TYPE SYUCOMM,  
\*      PBO  
.  
  PRIVATE SECTION.  
    DATA:  
        R\_DBIO TYPE REF TO LCL\_DB\_IO,  
        R\_PERSIST\_DB TYPE REF TO LCL\_PERSIST,  
        R\_TREE   TYPE REF TO CL\_SALV\_TREE,  
        R\_EVENTS TYPE REF TO CL\_SALV\_EVENTS\_TREE,  
        R\_FUNCTIONS TYPE REF TO CL\_SALV\_FUNCTIONS\_TREE,  
        R\_TOOLBAR TYPE REF TO CL\_GUI\_TOOLBAR,  
        R\_COLUMNS TYPE REF TO CL\_SALV\_COLUMNS,  
        R\_COLUMN TYPE REF TO CL\_SALV\_COLUMN,  
        R\_DSPSET TYPE REF TO CL\_SALV\_DISPLAY\_SETTINGS,  
        R\_LAYOUT TYPE REF TO CL\_SALV\_LAYOUT,  
        R\_SELECTIONS TYPE REF TO CL\_SALV\_SELECTIONS,  
        R\_SETTINGS TYPE REF TO CL\_SALV\_TREE\_SETTINGS,  
        R\_EVENT\_TAB TYPE REF TO CL\_SALV\_EVENTS\_TABLE,  
        R\_NODES TYPE REF TO CL\_SALV\_NODES,  
        R\_NODE TYPE REF TO CL\_SALV\_NODE,  
        R\_ITEM  TYPE REF TO CL\_SALV\_ITEM,  
        CUSTOM\_CONTAINER TYPE REF TO CL\_GUI\_CUSTOM\_CONTAINER,  
        T\_TAB TYPE TABLE OF \_OUTPUT,  
        T\_OPTREE TYPE TABLE OF \_OUTPUT.  
  
    METHODS:  
            ACCESS\_DATA,  
            ADD\_LEVEL IMPORTING LI\_LINE TYPE \_OUTPUT  
                                LI\_KEY TYPE LVC\_NKEY  
                      EXPORTING LO\_KEY TYPE LVC\_NKEY,  
            BUILD\_TREE,  
            CHECK\_FOR\_ERRORS IMPORTING T\_ERRTAB TYPE BAPIRET2\_T  
                             RETURNING VALUE(RVAL) TYPE BOOLEAN,  
            CHECK\_VALUES     IMPORTING I\_STR TYPE ANY  
                             RETURNING VALUE(RVAL) TYPE BOOLEAN,  
            CONV\_ELEMENTS    CHANGING STR1 TYPE ANY,  
            CREATE\_METAINF,  
            CREATE\_TREE,  
            CLS\_EVENT,  
            DB\_CALL,  
            DISP\_TREE,  
            EDIT\_COLUMNS,  
            FILTER\_WBS\_ELEMS IMPORTING I\_STR1 TYPE ANY  
                                       I\_STR2 TYPE ANY  
                             EXPORTING E\_LVL TYPE CHAR3  
                             RETURNING VALUE(RVAL) TYPE BOOLEAN,  
            FORMAT\_DATA,  
            GET\_FUNCTIONS,  
            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,  
            ON\_DOUBLE\_CLICK FOR EVENT IF\_SALV\_EVENTS\_TREE~DOUBLE\_CLICK  
              OF CL\_SALV\_EVENTS\_TREE  
                             IMPORTING COLUMNNAME  
                                       NODE\_KEY,  
            ON\_LINK\_CLICK FOR EVENT IF\_SALV\_EVENTS\_TREE~LINK\_CLICK  
              OF CL\_SALV\_EVENTS\_TREE  
                             IMPORTING COLUMNNAME  
                                       NODE\_KEY,  
  
           ON\_BUTTON\_CLICK FOR EVENT IF\_SALV\_EVENTS\_FUNCTIONS~AFTER\_SALV\_FUNCTION  
                OF CL\_SALV\_EVENTS\_TREE  
                             IMPORTING E\_SALV\_FUNCTION,  
          REG\_EVENTS,  
          SUPPLY\_DAT,  
          SORT\_TABLE,  
          TBL\_CONT\_MSGS.  
    CLASS-METHODS: CREATE\_CSV\_OP RETURNING VALUE(RVAL) TYPE BOOLEAN,  
                   SAVE\_BFILE,  
                   SAVE\_FFILE.  
    CLASS-DATA: V\_RECORDS TYPE I,  
                T\_FILE  TYPE TABLE OF \_OUTPUT,  
                T\_RESULT TYPE TABLE OF SOLISTI1,  
                T\_LOG TYPE TABLE OF BAPIRET2.  
ENDCLASS.                    "LCL\_CNTLR DEFINITION  
\*----------------------------------------------------------------------\*  
\*       CLASS LCL\_CNTLR IMPLEMENTATION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
CLASS LCL\_CNTLR IMPLEMENTATION.  
  METHOD CONSTRUCTOR.  
    CREATE OBJECT R\_DBIO.  
    R\_DBIO->VBUKR  = VBUKR.  
    R\_DBIO->VKOKR  = VKOKR.  
    R\_DBIO->STAT   = STAT.  
    R\_DBIO->PSPID  = PSPID.  
    R\_DBIO->POSID  = POSID.  
    R\_DBIO->POSKI  = POSKI.  
    R\_DBIO->PRCTR  = PRCTR.  
    R\_DBIO->PRART  = PRART.  
    ME->EXPATH = EXPATH.  
    ME->BTPATH = BTPATH.  
    ME->DWNLFF = DWNLFF.  
    ME->DWNLBT = DWNLBT.  
    IF ME->DWNLFF IS NOT INITIAL.  
      CLEAR ME->BTPATH.  
    ELSEIF ME->DWNLBT IS NOT INITIAL.  
      CLEAR ME->EXPATH.  
    ENDIF.  
  ENDMETHOD.                    "CONSTRUCTOR  
  METHOD ACCESS\_DATA.  
    DB\_CALL( ).  
    SORT\_TABLE(  ).  
    V\_ERROR\_CHECK = CHECK\_FOR\_ERRORS( T\_ERRTAB = T\_LOG ).  
  ENDMETHOD.                    "ACCESS\_DATA  
  METHOD ADD\_LEVEL.  
    DATA: 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.  
          IF LS\_OUTPUT-SUBTOT = 1.  
            LV\_TEXT = TEXT-F00.  
          ELSEIF LS\_OUTPUT-SUBTOT = 2.  
            LV\_TEXT = SPACE.  
          ELSEIF LS\_OUTPUT-SUBTOT IS INITIAL.  
            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.  
        ENDIF.  
        R\_NODE->SET\_TEXT( LV\_TEXT ).  
        R\_NODE->SET\_DATA\_ROW( LS\_OUTPUT ).  
        IF LS\_OUTPUT-PSPNR IS INITIAL OR LS\_OUTPUT-OBJNR IS INITIAL.  
          R\_NODE->SET\_ROW\_STYLE( IF\_SALV\_C\_TREE\_STYLE=>EMPHASIZED ).  
        ELSE.  
          IF LS\_OUTPUT-ACTL\_COST IS NOT INITIAL.  
            R\_ITEM = R\_NODE->GET\_ITEM( 'ACTL\_COST' ).  
            R\_ITEM->SET\_TYPE( IF\_SALV\_C\_ITEM\_TYPE=>LINK ).  
          ENDIF.  
          IF LS\_OUTPUT-ORDR\_COST IS NOT INITIAL.  
            R\_ITEM = R\_NODE->GET\_ITEM( 'ORDR\_COST' ).  
            R\_ITEM->SET\_TYPE( IF\_SALV\_C\_ITEM\_TYPE=>LINK ).  
          ENDIF.  
        ENDIF.  
        LO\_KEY = R\_NODE->GET\_KEY( ).  
      CATCH CX\_SALV\_MSG.  
        V\_ERROR\_CHECK = ABAP\_TRUE.  
        EXIT.  
    ENDTRY.  
  ENDMETHOD.                    "ADD\_LEVEL  
  METHOD BUILD\_TREE.  
    EDIT\_COLUMNS( ).  
    CREATE\_METAINF( ).  
    GET\_FUNCTIONS( ).  
    R\_TREE->DISPLAY( ).  
  ENDMETHOD.                    "BUILD\_TREE  
  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.                    "CHECK\_FOR\_ERRORS  
  METHOD CLS\_EVENT.  
  ENDMETHOD.                    "cls\_event  
  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.                    "CONV\_ELEMENTS  
  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.                    "CHECK\_VALUES  
  METHOD CHECK\_BFILE.  
    IF SY-BATCH IS NOT INITIAL.  
      IF BTPATH IS INITIAL.  
        RVAL = ABAP\_TRUE.  
        MESSAGE E000(KN) WITH 'Enter Valid Path' .  
      ELSE.  
        FILE = BTPATH.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.  
  
  METHOD CHECK\_FFILE.  
    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\_EXT     TYPE STRING.     " File Extension  
  
    CHECK EXPATH IS NOT INITIAL.  
    CALL FUNCTION 'SO\_SPLIT\_FILE\_AND\_PATH'  
      EXPORTING  
        FULL\_NAME     = EXPATH  
      IMPORTING  
        STRIPPED\_NAME = LV\_FNAME  
        FILE\_PATH     = LV\_DIR  
      EXCEPTIONS  
        X\_ERROR       = 1  
        OTHERS        = 2.  
    IF SY-SUBRC <> 0.  
      RVAL = ABAP\_TRUE.  
      MESSAGE E000(KN) WITH 'Invalid path: ' EXPATH.  
      EXIT.  
    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.  
        RVAL = ABAP\_TRUE.  
        MESSAGE E000(KN) WITH 'Invalid path: ' LV\_DIR.  
        EXIT.  
      ELSE.  
        TRANSLATE LV\_FNAME TO UPPER CASE.                 "#EC SYNTCHAR  
        SPLIT LV\_FNAME AT '.' INTO LV\_FNAME1 LV\_EXT.  
        IF LV\_EXT IS INITIAL.  
          V\_ERROR\_CHECK = ABAP\_TRUE.  
          MESSAGE E000(KN) WITH 'Invalid file extension '.  
          EXIT.  
  
        ELSE.  
          IF LV\_EXT CP C\_CSV OR LV\_EXT EQ C\_CSV2.  
  
          ELSE.  
            RVAL = ABAP\_TRUE.  
            MESSAGE E000(KN) WITH 'Invalid file extension '.  
            EXIT.  
          ENDIF.  
        ENDIF.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.                    "CHECK\_FFILE  
  METHOD CREATE\_CSV\_OP.  
    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,  
    LV\_PROJ\_POST1    TYPE STRING,  
    LV\_PRPS\_POST1    TYPE STRING,  
    LV\_PRPS\_USR00    TYPE STRING.  
    FIELD-SYMBOLS:  
      <FS\_RESULT> TYPE SOLISTI1,  
      <FS\_OUTPUT> TYPE \_OUTPUT.  
  
    IF T\_OUTPUT IS NOT INITIAL.  
      CLEAR T\_RESULT[].  
      APPEND INITIAL LINE TO T\_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-C34"'Task Date'  
        INTO <FS\_RESULT>-LINE SEPARATED BY C\_SEP.  
      LOOP AT T\_OUTPUT 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.  
        IF <FS\_OUTPUT>-PROJ\_POST1 CS C\_SEP.  
          CONCATENATE C\_QUOTE  <FS\_OUTPUT>-PROJ\_POST1 C\_QUOTE INTO LV\_PROJ\_POST1.  
        ELSE.  
          LV\_PROJ\_POST1 =  <FS\_OUTPUT>-PROJ\_POST1.  
        ENDIF.  
        IF <FS\_OUTPUT>-PRPS\_POST1 CS C\_SEP.  
          CONCATENATE C\_QUOTE  <FS\_OUTPUT>-PRPS\_POST1 C\_QUOTE INTO LV\_PRPS\_POST1.  
        ELSE.  
          LV\_PRPS\_POST1 = <FS\_OUTPUT>-PRPS\_POST1.  
        ENDIF.  
        IF <FS\_OUTPUT>-PRPS\_USR00 CS C\_SEP.  
          CONCATENATE C\_QUOTE  <FS\_OUTPUT>-PRPS\_USR00 C\_QUOTE INTO LV\_PRPS\_USR00.  
        ELSE.  
          LV\_PRPS\_USR00 = <FS\_OUTPUT>-PRPS\_USR00.  
        ENDIF.  
        APPEND INITIAL LINE TO T\_RESULT ASSIGNING <FS\_RESULT>.  
        CONCATENATE <FS\_OUTPUT>-PROJ\_PSPID"'Project'  
                    LV\_PROJ\_POST1"'Project Description'  
                    <FS\_OUTPUT>-POSID" 'Task'  
                    LV\_PRPS\_POST1"'Task Description'  
                    <FS\_OUTPUT>-PARVW"'Partner Function'  
                    <FS\_OUTPUT>-KUNNR"'Customer'  
                    LV\_PRPS\_USR00"'Department Total Code'  
                    LV\_ORIG\_COST"'Original Cost'  
                    LV\_CURR\_BUDG"'Current Budget'  
                    LV\_ORDR\_COST"'Ordered Cost'  
                    LV\_ACTL\_COST"'Actual Cost'  
                    LV\_END\_COST"'Projected End Cost'  
                    <FS\_OUTPUT>-PRCTR"'Division'  
                    SY-DATUM"'Snapshot Date'  
                    <FS\_OUTPUT>-PROJ\_STAT"Project Status'  
                    <FS\_OUTPUT>-TASK\_STAT"'Task Status'  
                    <FS\_OUTPUT>-TASK\_DATE "'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,  
          LV\_PROJ\_POST1,  
          LV\_PRPS\_POST1,  
          LV\_PRPS\_USR00.  
      ENDLOOP.  
    ELSE.  
      RVAL = V\_ERROR\_CHECK = ABAP\_TRUE.  
      EXIT.  
    ENDIF.  
  ENDMETHOD.                    "CREATE\_CSV\_HDR  
  METHOD CREATE\_CONTROLS.  
\*    IF ME->CUSTOM\_CONTAINER IS NOT BOUND.  
\*    IF CL\_SALV\_TREE=>IS\_OFFLINE( ) EQ IF\_SALV\_C\_BOOL\_SAP=>FALSE.  
\*      CREATE OBJECT ME->CUSTOM\_CONTAINER  
\*        EXPORTING  
\*          CONTAINER\_NAME = 'CONTAINER'.  
\*    ENDIF.  
\*    ENDIF.  
\*    ME->INIT\_ALV( ).  
  ENDMETHOD.                    "CREATE\_CONTROLS  
  
  METHOD CREATE\_TREE.  
    TRY.  
        CL\_SALV\_TREE=>FACTORY(  
\*          EXPORTING  
\*            R\_CONTAINER  = ME->CUSTOM\_CONTAINER  
          IMPORTING  
            R\_SALV\_TREE = R\_TREE  
          CHANGING  
            T\_TABLE      =  T\_TAB ).  
      CATCH CX\_SALV\_NO\_NEW\_DATA\_ALLOWED  
            CX\_SALV\_ERROR.  
        V\_ERROR\_CHECK = ABAP\_TRUE.  
        EXIT.  
    ENDTRY.  
    R\_TREE->SET\_SCREEN\_STATUS(  
  PFSTATUS      =  'SALV\_STANDARD'  
  REPORT        =  'SALV\_DEMO\_TREE\_EVENTS'  
  SET\_FUNCTIONS =  R\_TREE->C\_FUNCTIONS\_ALL ).  
  ENDMETHOD.                    "CREATE\_TREE  
  METHOD CREATE\_METAINF.  
    DATA: 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.  
    R\_SETTINGS = R\_TREE->GET\_TREE\_SETTINGS( ).  
    R\_SETTINGS->SET\_HIERARCHY\_HEADER( TEXT-HD1 ).  
    R\_SETTINGS->SET\_HEADER( TEXT-HD2 ).  
    R\_SETTINGS->SET\_HIERARCHY\_TOOLTIP( TEXT-HT1 ).  
    R\_SETTINGS->SET\_HIERARCHY\_SIZE( 40 ).  
    R\_SETTINGS->SET\_HIERARCHY\_ICON( '@3Q@' ).  
  
  
    READ TABLE T\_OPTREE INTO LS\_OUTPUT INDEX 1.  
  
    CREATE OBJECT LR\_GRID.  
  
    CREATE OBJECT LR\_GRID.  
    LR\_GRID->CREATE\_GRID( ROW    = 2  
                          COLUMN = 2 ).  
    LR\_GRID->CREATE\_HEADER\_INFORMATION( ROW     = 1  
                                        COLUMN  = 2  
                                        TEXT    = R\_DBIO->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' ).  
  
  
    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     = 2  
                                      COLUMN  = 2  
                                      TEXT    = LV\_TEXT1  
                                      TOOLTIP = TEXT-H02 ).  
    LR\_LABEL = LR\_GRID\_1->CREATE\_LABEL( ROW     = 2  
                                        COLUMN  = 3  
                                        TEXT    = TEXT-H01 ).  
    LR\_TEXT = LR\_GRID\_1->CREATE\_TEXT( ROW     = 2  
                                       COLUMN  = 4  
                                       TEXT    = SY-TITLE  
                                       TOOLTIP = TEXT-H01 ).  
  
    LR\_LABEL = LR\_GRID\_1->CREATE\_LABEL( ROW     = 3  
                                       COLUMN  = 1  
                                        TEXT    = TEXT-H03 ).  
    LR\_TEXT = LR\_GRID\_1->CREATE\_TEXT( ROW     = 3  
                                      COLUMN  = 2  
                                      TEXT    = SY-UNAME  
                                      TOOLTIP = TEXT-H03 ).  
    LR\_LABEL = LR\_GRID\_1->CREATE\_LABEL( ROW     = 3  
                                        COLUMN  = 3  
                                        TEXT    = TEXT-H04 ).  
    LR\_TEXT = LR\_GRID\_1->CREATE\_TEXT( ROW     = 3  
                                      COLUMN  = 4  
                                      TEXT    = LS\_OUTPUT-PROJ\_PSPID  
                                      TOOLTIP = TEXT-H05 ).  
    LR\_LABEL = LR\_GRID\_1->CREATE\_LABEL( ROW     = 4  
                                        COLUMN  = 1  
                                        TEXT    = TEXT-H06 ).  
    LR\_TEXT = LR\_GRID\_1->CREATE\_TEXT( ROW     = 4  
                                      COLUMN  = 2  
                                      TEXT    = LS\_OUTPUT-PROJ\_TYPE  
                                      TOOLTIP = TEXT-H06 ).  
    LR\_LABEL = LR\_GRID\_1->CREATE\_LABEL( ROW     = 4  
                                        COLUMN  = 3  
                                        TEXT    = TEXT-H08 ).  
    LR\_TEXT = LR\_GRID\_1->CREATE\_TEXT( ROW     = 4  
                                      COLUMN  = 4  
                                      TEXT    = LS\_OUTPUT-PROJ\_STAT  
                                      TOOLTIP = TEXT-H08 ).  
  
    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     = 3  
                                        COLUMN  = 5  
                                        TEXT    = TEXT-H07  ).  
  
    LR\_TEXT = LR\_GRID\_1->CREATE\_TEXT( ROW     = 3  
                                      COLUMN  = 6  
                                      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     =     6  
        COLUMN  =     3  
    ).  
    CLEAR: LV\_TEXT1, LV\_LINES.  
    LV\_LINES = LV\_LINES + 3.  
    DO 5 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(  
                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.                    "CREATE\_METAINF  
  METHOD DISP\_TREE.  
    CREATE\_TREE( ).  
    SUPPLY\_DAT( ).  
    REG\_EVENTS( ).  
    BUILD\_TREE( ).  
  ENDMETHOD.                    "DISP\_TREE  
  METHOD DWNLD\_FILE.  
    DATA:  
      V\_TIME\_STAMP(14),  
      V\_PRG\_ID(22),  
      V\_PRG\_ID2(22),  
      V\_CNT TYPE I,  
      V\_IDX TYPE I,  
      LV\_PHYSICAL\_PATH(255),  
      LV\_PHYSICAL\_PATH\_B(255),  
      WA\_RETURN  TYPE BAPIRET2.  
  
  
    CONCATENATE SY-DATUM  
                SY-UZEIT  
           INTO V\_TIME\_STAMP.  
  ENDMETHOD.  
  METHOD EDIT\_COLUMNS.  
  
    TRY .  
        R\_COLUMNS = R\_TREE->GET\_COLUMNS( ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'STEP\_NAME' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C33 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C33 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C33 ).  
        R\_COLUMN->SET\_OUTPUT\_LENGTH( '35' ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PROJ\_PLSEZ' ).  
        R\_COLUMN->SET\_OPTIMIZED( ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C02 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C03 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C04 ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'SLS\_PRICE' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C05 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C05 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C06 ).  
        R\_COLUMN->SET\_OUTPUT\_LENGTH( '20' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN->SET\_TECHNICAL( IF\_SALV\_C\_BOOL\_SAP=>TRUE  ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'SLS\_PRICE\_C' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C05 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C05 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C06 ).  
        R\_COLUMN->SET\_OUTPUT\_LENGTH( '20' ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'DEPT' ).  
        R\_COLUMN->SET\_OPTIMIZED( ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C07 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C07 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C07 ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'ORIG\_COST' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C11 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C12 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C12 ).  
        R\_COLUMN->SET\_OUTPUT\_LENGTH( '20' ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'CURR\_BUDG' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C13 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C14 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C14  ).  
        R\_COLUMN->SET\_OUTPUT\_LENGTH( '20' ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'END\_COST' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C15  ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C16  ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C16 ).  
        R\_COLUMN->SET\_OUTPUT\_LENGTH( '20' ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'ORDR\_COST' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C17 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C17 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C18 ).  
        R\_COLUMN->SET\_OUTPUT\_LENGTH( '20' ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'ACTL\_COST' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C19 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C19  ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C20 ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'ORDR\_PERC' ).  
        R\_COLUMN->SET\_ALIGNMENT( CL\_SALV\_COLUMN=>RIGHT ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C21 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C21 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C21 ).  
        R\_COLUMN->SET\_OUTPUT\_LENGTH( '20' ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'CURR\_ACTL' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C22 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C23 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C24 ).  
        R\_COLUMN->SET\_OUTPUT\_LENGTH( '20' ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'TASK\_STAT' ).  
        R\_COLUMN->SET\_OUTPUT\_LENGTH( '6' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C25 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C25 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C25  ).  
  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PROJ\_TYPE' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE  ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'COMP\_NAME' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE  ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'STATUS' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE  ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PROJ\_STAT' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE  ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'TASK\_DATE' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE  ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'VBUKR' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE  ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'VKOKR' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE  ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PSPNR' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE  ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PRPS\_POSKI' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PRCTR ' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PSPHI' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'OBJNR' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'KUNNR' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PARVW' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PRPS\_USR00' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C07 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C07 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C07 ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'LEVEL' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN->SET\_TECHNICAL( IF\_SALV\_C\_BOOL\_SAP=>TRUE ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PROJ\_PSPID' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C01 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C01 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C01 ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PROJ\_POST1' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-H05 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-H05 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-H05 ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'POSID' ).  
        R\_COLUMN->SET\_LONG\_TEXT( TEXT-C08 ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT( TEXT-C08 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C08 ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'PRPS\_POST1' ).  
        R\_COLUMN->SET\_OPTIMIZED( ).  
        R\_COLUMN->SET\_LONG\_TEXT(  TEXT-C09  ).  
        R\_COLUMN->SET\_MEDIUM\_TEXT(  TEXT-C10 ).  
        R\_COLUMN->SET\_SHORT\_TEXT( TEXT-C10 ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN = R\_COLUMNS->GET\_COLUMN( COLUMNNAME = 'SUBTOT' ).  
        R\_COLUMN->SET\_VISIBLE( IF\_SALV\_C\_BOOL\_SAP=>FALSE ).  
        R\_COLUMN->SET\_TECHNICAL( IF\_SALV\_C\_BOOL\_SAP=>TRUE ).  
      CATCH CX\_SALV\_NOT\_FOUND CX\_SALV\_DATA\_ERROR.  
        V\_ERROR\_CHECK = ABAP\_TRUE.  
        EXIT.  
    ENDTRY.  
  ENDMETHOD.                    "EDIT\_COLUMNS  
  METHOD ERROR\_CHECK.  
    RVAL = V\_ERROR\_CHECK.  
  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 CS C\_CSV2.  
        ELSE.  
          CONCATENATE EXPATH C\_CSV INTO EXPATH.  
        ENDIF.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.                                                "F4\_HLPFILE  
  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\_DBIO->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.  
        SPLIT V\_POS3 AT C\_PERIOD INTO V\_POS1 E\_LVL.  
        READ TABLE ME->R\_DBIO->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.  
      ELSE.  
        RVAL = ABAP\_FALSE.  
      ENDIF.  
    ELSE.  
      RVAL = ABAP\_FALSE.  
    ENDIF.  
  
  ENDMETHOD.                    "FILTER\_WBS\_ELEMS  
  METHOD GET\_FUNCTIONS.  
    TRY .  
        R\_FUNCTIONS = R\_TREE->GET\_FUNCTIONS( ).  
      CATCH CX\_SALV\_MSG.  
        V\_ERROR\_CHECK = ABAP\_TRUE.  
        EXIT.  
    ENDTRY.  
    R\_FUNCTIONS->SET\_GROUP\_AGGREGATION( ABAP\_FALSE ).  
  
  
  ENDMETHOD.                    "GET\_FUNCTIONS  
  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\_DBIO->T\_VBAP  
      ASSIGNING <VBAP>  
      WITH KEY PS\_PSP\_PNR = S\_STR-PSPNR  
      BINARY SEARCH.  
    IF SY-SUBRC = 0.  
      READ TABLE ME->R\_DBIO->T\_VBPA  
        ASSIGNING <VBPA>  
        WITH KEY VBELN = <VBAP>-VBELN  
        BINARY SEARCH.  
      IF SY-SUBRC = 0.  
        READ TABLE ME->R\_DBIO->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\_DBIO->T\_KNA1  
          ASSIGNING <KNA1> WITH KEY KUNNR = <VBPA>-KUNNR  
          BINARY SEARCH.  
        IF SY-SUBRC = 0.  
          E\_KUNNR = <KNA1>-NAME1.  
        ENDIF.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.                    "GET\_CUSTDATA  
  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\_DBIO->T\_COEP  
         ASSIGNING <COEP>  
         WHERE OBJNR = LV\_OBJNR.  
      <S\_STR>-ACTL\_COST = <S\_STR>-ACTL\_COST + <COEP>-WTGBTR.  
    ENDLOOP.  
  
  
  ENDMETHOD.                    "GET\_ACTUALS  
  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\_DBIO->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.                    "GET\_ORDERED  
  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\_DBIO->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\_DBIO->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.  
          READ TABLE ME->R\_DBIO->T\_TSTAT  
            ASSIGNING <FS\_TASK> WITH KEY OBJNR = I\_TASK  
                                         STAT  = C\_I0002 BINARY SEARCH.  
          IF SY-SUBRC = 0 AND <FS\_TASK>-INACT IS INITIAL.  
            LV\_STAT = C\_I0002.  
          ELSE.  
  
          ENDIF.  
        ENDIF.  
      ENDIF.  
      IF LV\_STAT IS NOT INITIAL.  
        READ TABLE ME->R\_DBIO->T\_TJ02T  
          ASSIGNING <TJ02T> WITH KEY ISTAT = LV\_STAT  
                                     SPRAS = SY-LANGU BINARY SEARCH.  
        IF SY-SUBRC = 0.  
          RVAL = <TJ02T>-TXT30.  
        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.  
      IF LV\_LINE IS NOT INITIAL.  
        RVAL = LV\_LINE.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.                    "GET\_STATTXT  
  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\_DBIO->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.                    "GET\_VERSION  
  METHOD ON\_LINK\_CLICK.  
    ON\_DOUBLE\_CLICK(  
      EXPORTING  
        COLUMNNAME = COLUMNNAME  
        NODE\_KEY   = NODE\_KEY  
    ).  
  ENDMETHOD.  
  METHOD ON\_DOUBLE\_CLICK.  
    DATA: LV\_REPID TYPE SY-REPID VALUE 'RKPEP000',  
          LV\_ITEM\_GROUP TYPE  KAEP\_ITEM\_GROUP,  
          LV\_REP\_OBJECT TYPE  KAEP\_REP\_OBJECT VALUE 'PD',  
          LR\_ALV\_GRID TYPE REF TO CL\_GUI\_ALV\_GRID,  
          LS\_OTREE TYPE \_OUTPUT,  
          LT\_SELECTION TYPE KAEP\_T\_COSEL,  
          LT\_PD\_SELECTION TYPE  KAEP\_T\_COSEL .  
  
    FIELD-SYMBOLS:<FS\_SELECTION> TYPE KAEP\_COSEL.  
  
    READ TABLE T\_OPTREE INTO LS\_OTREE INDEX NODE\_KEY.  
    IF SY-SUBRC = 0.  
      IF LS\_OTREE-OBJNR IS NOT INITIAL.  
        CASE COLUMNNAME.  
          WHEN '&Hierarchy'.  
            SET PARAMETER ID 'PSP' FIELD LS\_OTREE-PROJ\_PSPID.  
            CALL TRANSACTION 'CJ2A' AND SKIP FIRST SCREEN. "#EC CI\_CALLTA.  
          WHEN 'ACTL\_COST'.  
            IF LS\_OTREE-ACTL\_COST IS INITIAL.  
              MESSAGE I422(KB).  
              RETURN.  
            ELSE.  
              LV\_ITEM\_GROUP = 'COAC'.  
              APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
              <FS\_SELECTION>-FIELD = TEXT-S01."GJAHR  
              <FS\_SELECTION>-SIGN = C\_I.  
              <FS\_SELECTION>-OPTION = C\_BT.  
              <FS\_SELECTION>-LOW = C\_0000.  
              <FS\_SELECTION>-HIGH = C\_2999.  
              APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
              <FS\_SELECTION>-FIELD = TEXT-S00."VALUTYP  
              <FS\_SELECTION>-SIGN = C\_I.  
              <FS\_SELECTION>-OPTION = C\_EQ.  
              <FS\_SELECTION>-LOW = C\_0.  
              APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
              <FS\_SELECTION>-FIELD = TEXT-S03."VERSN  
              <FS\_SELECTION>-SIGN = C\_I.  
              <FS\_SELECTION>-OPTION = C\_EQ.  
              <FS\_SELECTION>-LOW = C\_ZERO.  
              APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
              <FS\_SELECTION>-FIELD = TEXT-S00."BELTP  
              <FS\_SELECTION>-SIGN = C\_I.  
              <FS\_SELECTION>-OPTION = C\_BT.  
              <FS\_SELECTION>-LOW = C\_0.  
              <FS\_SELECTION>-HIGH = C\_1.  
              APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
              <FS\_SELECTION>-FIELD = TEXT-S04."WRTTP.  
              <FS\_SELECTION>-SIGN = C\_I.  
              <FS\_SELECTION>-OPTION = C\_EQ.  
              <FS\_SELECTION>-LOW = C\_04.  
              APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
              <FS\_SELECTION>-FIELD = TEXT-S04."WRTTP.  
              <FS\_SELECTION>-SIGN = C\_I.  
              <FS\_SELECTION>-OPTION = C\_EQ.  
              <FS\_SELECTION>-LOW = C\_11.  
              IF LS\_OTREE-LEVEL = 1 OR LS\_OTREE-LEVEL = 2.  
                LOOP AT T\_OPTREE INTO LS\_OTREE FROM NODE\_KEY  
                    WHERE OBJNR IS NOT INITIAL AND LEVEL > 2  
                      AND ACTL\_COST IS NOT INITIAL.  
                  APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
                  <FS\_SELECTION>-FIELD = TEXT-S05."OBJNR  
                  <FS\_SELECTION>-SIGN = C\_I.  
                  <FS\_SELECTION>-OPTION = C\_EQ.  
                  <FS\_SELECTION>-LOW = LS\_OTREE-OBJNR.  
                ENDLOOP.  
              ELSE.  
                APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
                <FS\_SELECTION>-FIELD = TEXT-S05."OBJNR  
                <FS\_SELECTION>-SIGN = C\_I.  
                <FS\_SELECTION>-OPTION = C\_EQ.  
                <FS\_SELECTION>-LOW = LS\_OTREE-OBJNR.  
              ENDIF.  
            ENDIF.  
          WHEN 'ORDR\_COST'.  
            IF LS\_OTREE-ORDR\_COST = LS\_OTREE-ACTL\_COST  
              OR LS\_OTREE-ORDR\_COST IS INITIAL.  
              MESSAGE I422(KB).  
              RETURN.  
            ELSE.  
              LV\_ITEM\_GROUP = 'CM'.  
              APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
              <FS\_SELECTION>-FIELD = TEXT-S01."GJAHR  
              <FS\_SELECTION>-SIGN = C\_I.  
              <FS\_SELECTION>-OPTION = C\_BT.  
              <FS\_SELECTION>-LOW = C\_0000.  
              <FS\_SELECTION>-HIGH = C\_2999.  
              APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
              <FS\_SELECTION>-FIELD = TEXT-S03."VERSN  
              <FS\_SELECTION>-SIGN = C\_I.  
              <FS\_SELECTION>-OPTION = C\_EQ.  
              <FS\_SELECTION>-LOW = C\_ZERO.  
              APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
              <FS\_SELECTION>-FIELD = TEXT-S04."WRTTP.  
              <FS\_SELECTION>-SIGN = C\_I.  
              <FS\_SELECTION>-OPTION = C\_EQ.  
              <FS\_SELECTION>-LOW = C\_21.  
              APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
              <FS\_SELECTION>-FIELD = TEXT-S04."WRTTP.  
              <FS\_SELECTION>-SIGN = C\_I.  
              <FS\_SELECTION>-OPTION = C\_EQ.  
              <FS\_SELECTION>-LOW = C\_22.  
              APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
              <FS\_SELECTION>-FIELD = TEXT-S04."WRTTP.  
              <FS\_SELECTION>-SIGN = C\_I.  
              <FS\_SELECTION>-OPTION = C\_EQ.  
              <FS\_SELECTION>-LOW = C\_24.  
              APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
              <FS\_SELECTION>-FIELD = TEXT-S04."WRTTP.  
              <FS\_SELECTION>-SIGN = C\_I.  
              <FS\_SELECTION>-OPTION = C\_EQ.  
              <FS\_SELECTION>-LOW = C\_26.  
              IF LS\_OTREE-LEVEL = 1 OR LS\_OTREE-LEVEL = 2.  
                LOOP AT T\_OPTREE INTO LS\_OTREE FROM NODE\_KEY  
                    WHERE OBJNR IS NOT INITIAL AND LEVEL > 2  
                      AND ORDR\_COST IS NOT INITIAL.  
                  APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
                  <FS\_SELECTION>-FIELD = TEXT-S05."OBJNR  
                  <FS\_SELECTION>-SIGN = C\_I.  
                  <FS\_SELECTION>-OPTION = C\_EQ.  
                  <FS\_SELECTION>-LOW = LS\_OTREE-OBJNR.  
                ENDLOOP.  
              ELSE.  
                APPEND INITIAL LINE TO LT\_SELECTION ASSIGNING <FS\_SELECTION>.  
                <FS\_SELECTION>-FIELD = TEXT-S05."OBJNR  
                <FS\_SELECTION>-SIGN = C\_I.  
                <FS\_SELECTION>-OPTION = C\_EQ.  
                <FS\_SELECTION>-LOW = LS\_OTREE-OBJNR.  
              ENDIF.  
            ENDIF.  
          WHEN OTHERS.  
            RETURN.  
        ENDCASE.  
        SORT LT\_SELECTION.  
        DELETE ADJACENT DUPLICATES FROM LT\_SELECTION COMPARING ALL FIELDS.  
        CHECK LT\_SELECTION[] IS NOT INITIAL.  
        CALL FUNCTION 'K\_LINE\_ITEMS\_SELECT\_AND\_LIST'  
          EXPORTING  
            I\_CALLING\_REPORT    = LV\_REPID  
            I\_ITEM\_GROUP        = LV\_ITEM\_GROUP  
            I\_REP\_OBJECT        = LV\_REP\_OBJECT  
\*           I\_DISPLAY\_VARIANT   =  
            I\_CO\_AREA           = ME->R\_DBIO->VKOKR  
            I\_ACTUAL\_VERSION    = C\_ZERO  
            IT\_SELECTION        = LT\_SELECTION  
            I\_SHOWNOZERO        = C\_X  
            I\_CALLED\_BY\_RRI     = C\_X  
            IT\_PD\_SELECTION     = LT\_PD\_SELECTION  
            I\_USEDB             = C\_X  
            I\_TRANSFER\_VERSIONS = C\_X  
            I\_USE\_GRID          = C\_X  
            I\_GROUP\_OBJ         = C\_X  
\*           TABLES  
\*           T\_ARSEL             =  
\*           CHANGING  
\*           IT\_KAEPX\_DATA\_CARRIER         =  
\*           ER\_ALV\_GRID         = lR\_ALV\_GRID  
          EXCEPTIONS  
            UNKNOWN\_ITEM\_GROUP  = 1  
            NO\_DATA\_SELECTED    = 2  
            OTHERS              = 3.  
        IF SY-SUBRC <> 0.  
\* Implement suitable error handling here  
          MESSAGE I422(KB).  
        ENDIF.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.                    "handle\_events  
  METHOD ON\_BUTTON\_CLICK.  
    CASE E\_SALV\_FUNCTION.  
      WHEN 'EXPORT'.  
        ME->SAVE\_FFILE( ).  
    ENDCASE.  
  ENDMETHOD.  
  METHOD INIT\_ALV.  
    ACCESS\_DATA( ).  
    CHECK ERROR\_CHECK( ) <> ABAP\_TRUE.  
    FORMAT\_DATA( ).  
    CHECK DWNLFF = ABAP\_FALSE AND DWNLBT = ABAP\_FALSE.  
    DISP\_TREE( ).  
  ENDMETHOD.                    "init\_alv  
  METHOD SORT\_TABLE.  
    SORT ME->R\_DBIO->T\_PSTAT BY OBJNR STAT.  
    SORT ME->R\_DBIO->T\_TSTAT BY OBJNR STAT.  
    SORT ME->R\_DBIO->T\_RPSCO BY OBJNR WRTTP GJAHR VORGA VERSN BELTP WLP00.  
    SORT ME->R\_DBIO->T\_TPART BY SPRAS PARVW.  
    SORT ME->R\_DBIO->T\_VBAP BY PS\_PSP\_PNR.  
    SORT ME->R\_DBIO->T\_VBPA BY VBELN.  
    SORT ME->R\_DBIO->T\_VBPA BY VBELN.  
    SORT ME->R\_DBIO->T\_TJ02T BY ISTAT SPRAS.  
    SORT ME->R\_DBIO->T\_COEP BY OBJNR.  
\*    SORT ME->R\_DBIO->T\_BPGE BY OBJNR WRTTP.  
    SORT ME->R\_DBIO->T\_COOI BY OBJNR.  
    SORT ME->R\_DBIO->T\_TCJ1T BY PRART.  
    SORT ME->R\_DBIO->T\_BPHI BY VERSN DESCENDING.  
  ENDMETHOD.                    "SORT\_TABLE  
  METHOD DB\_CALL.  
    CREATE OBJECT R\_PERSIST\_DB.  
    R\_PERSIST\_DB->FETCH\_DATA( CHANGING R\_DBIO = R\_DBIO ).  
  ENDMETHOD.                    "DB\_CALL  
  METHOD TBL\_CONT\_MSGS.  
    IF R\_DBIO->T\_PRPS IS INITIAL.  
  
    ENDIF.  
    IF R\_DBIO->T\_PROJ IS INITIAL.  
  
    ENDIF.  
  ENDMETHOD.                    "TBL\_CONT\_MSGS  
  METHOD FORMAT\_DATA.  
  
    DATA:  
      LV\_KEEP\_RECORD   TYPE BOOLEAN,  
      LV\_POSID         TYPE PS\_POSID,  
      LV\_STATUS        TYPE J\_STATUS,  
      LV\_PRJSTA        TYPE J\_STATUS,  
      LV\_ORD\_COST      TYPE BP\_WPL,  
      LS\_LVL\_ONE       TYPE \_OUTPUT,  
      LS\_LVL\_TWO       TYPE \_OUTPUT,  
      LS\_LVL\_ZERO      TYPE \_OUTPUT,  
      LV\_VERSION       TYPE BP\_VERSION,  
      LV\_LVL(3)        TYPE C.  
    DATA: LT\_SUBTOT TYPE TABLE OF \_DEPTOT.  
    DATA: LS\_SUBTOT TYPE \_DEPTOT,  
          LS\_OPTREE TYPE \_OUTPUT.  
  
    FIELD-SYMBOLS:  
      <FS\_PRPS>    LIKE LINE OF R\_DBIO->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\_DBIO->T\_TCJ1T,  
      <FS\_PROJ>    LIKE LINE OF R\_DBIO->T\_PROJ.  
  
    LOOP AT ME->R\_DBIO->T\_PROJ ASSIGNING <FS\_PROJ>.  
      CLEAR:  
        LV\_KEEP\_RECORD,  
        LS\_LVL\_ZERO,  
        LS\_LVL\_ONE,  
        LS\_LVL\_TWO,  
        LT\_SUBTOT,  
        LV\_LVL.  
      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\_DBIO->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 .  
      LV\_PRJSTA = GET\_STATTXT( I\_PROJ = <FS\_PROJ>-OBJNR ).  
      <FS\_OPTREE>-TASK\_STAT = <FS\_OPTREE>-PROJ\_STAT = LV\_PRJSTA(1).  
      LS\_LVL\_ZERO = <FS\_OPTREE>.  
      LOOP AT ME->R\_DBIO->T\_PRPS ASSIGNING <FS\_PRPS>  
           WHERE PSPHI = <FS\_PROJ>-PSPNR.  
        LV\_KEEP\_RECORD = FILTER\_WBS\_ELEMS( EXPORTING  
                                          I\_STR1 = <FS\_PRPS>  
                                          I\_STR2 = <FS\_PROJ>  
                                          IMPORTING  
                                           E\_LVL = LV\_LVL  ).  
        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.  
            LS\_LVL\_ONE-OBJNR = <FS\_PRPS>-OBJNR.  
            LS\_LVL\_ONE-COMP\_NAME = ME->R\_DBIO->COMP\_NAME.  
          ENDIF.  
\*8888888888888 SUBTOTAL 700  
          IF LV\_LVL > 700 AND LS\_LVL\_TWO IS INITIAL.  
            APPEND INITIAL LINE TO T\_OPTREE ASSIGNING <FS\_OPTREE>.  
            MOVE-CORRESPONDING LS\_LVL\_ONE TO <FS\_OPTREE>.  
            <FS\_OPTREE>-STEP\_NAME = TEXT-F07.  
            <FS\_OPTREE>-LEVEL = 3.  
            CLEAR <FS\_OPTREE>-POSID.  
            CLEAR <FS\_OPTREE>-OBJNR.  
            CLEAR <FS\_OPTREE>-PROJ\_POST1.  
            CLEAR <FS\_OPTREE>-PROJ\_PLSEZ.  
            CLEAR <FS\_OPTREE>-PROJ\_STAT.  
            CLEAR <FS\_OPTREE>-TASK\_STAT.  
            CLEAR <FS\_OPTREE>-SLS\_PRICE\_C.  
            CLEAR <FS\_OPTREE>-SLS\_PRICE.  
  
            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\_OPTREE>-ORDR\_PERC DECIMALS 2 RIGHT-JUSTIFIED.  
            ELSE.  
              WRITE LV\_ORD\_COST TO <FS\_OPTREE>-ORDR\_PERC DECIMALS 2 RIGHT-JUSTIFIED.  
            ENDIF.  
            LS\_LVL\_TWO = <FS\_OPTREE>.  
          ENDIF.  
\*8888888888888 SUBTOTAL 700  
          APPEND INITIAL LINE TO T\_OPTREE ASSIGNING <FS\_OPTREE>.  
          APPEND INITIAL LINE TO T\_OUTPUT ASSIGNING <FS\_OUTPUT>.  
          READ TABLE ME->R\_DBIO->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\_DBIO->COMP\_NAME.  
          CLEAR LV\_STATUS.  
          LV\_STATUS = GET\_STATTXT( I\_TASK = <FS\_PRPS>-OBJNR ).  
          <FS\_OUTPUT>-TASK\_STAT = LV\_STATUS(1).  
          <FS\_OUTPUT>-PROJ\_STAT = LV\_PRJSTA(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\_DBIO->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.  
  
            WRITE <FS\_RPSCO>-WLP00 TO LS\_LVL\_ONE-SLS\_PRICE\_C  
             STYLE 1 DECIMALS 2 RIGHT-JUSTIFIED .  
          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\_DBIO->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\_DBIO->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\_DBIO->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 = R\_DBIO->VBUKR.  
          <FS\_OUTPUT>-VKOKR = R\_DBIO->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>-OBJNR = <FS\_PRPS>-OBJNR.  
          IF <FS\_PRPS>-STUFE = C\_1.  
            <FS\_OPTREE>-LEVEL = 2.  
          ELSE.  
            <FS\_OPTREE>-LEVEL = 3.  
            <FS\_OPTREE>-STEP\_NAME = <FS\_OUTPUT>-PRPS\_POST1 .  
            CLEAR <FS\_OPTREE>-PROJ\_POST1.  
            CLEAR <FS\_OPTREE>-PROJ\_PLSEZ.  
  
            MOVE-CORRESPONDING <FS\_OUTPUT> TO LS\_SUBTOT.  
            CLEAR LS\_SUBTOT-ORDR\_PERC.  
            COLLECT LS\_SUBTOT INTO LT\_SUBTOT.  
            CLEAR LS\_SUBTOT.  
          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.  
              ENDIF.  
              UNASSIGN <FS\_OPTREE>.  
              READ TABLE T\_OPTREE ASSIGNING <FS\_OPTREE>  
                WITH KEY POSID = LV\_POSID.  
              IF SY-SUBRC = 0.  
                DELETE T\_OPTREE INDEX SY-TABIX.  
              ENDIF.  
            ENDIF.  
          ENDIF.  
        ELSE.  
          CONTINUE.  
        ENDIF.  
      ENDLOOP.  
\*8888888888888 SUBTOTAL  
      APPEND INITIAL LINE TO T\_OPTREE ASSIGNING <FS\_OPTREE>.  
      MOVE-CORRESPONDING LS\_LVL\_ONE TO <FS\_OPTREE>.  
      <FS\_OPTREE>-STEP\_NAME = TEXT-F08.  
      <FS\_OPTREE>-ORIG\_COST = LS\_LVL\_ONE-ORIG\_COST - LS\_LVL\_TWO-ORIG\_COST.  
      <FS\_OPTREE>-CURR\_BUDG = LS\_LVL\_ONE-CURR\_BUDG - LS\_LVL\_TWO-CURR\_BUDG.  
      <FS\_OPTREE>-END\_COST  = LS\_LVL\_ONE-END\_COST - LS\_LVL\_TWO-CURR\_BUDG.  
      <FS\_OPTREE>-ACTL\_COST = LS\_LVL\_ONE-ACTL\_COST - LS\_LVL\_TWO-ACTL\_COST.  
      <FS\_OPTREE>-ORDR\_COST = LS\_LVL\_ONE-ORDR\_COST - LS\_LVL\_TWO-ORDR\_COST.  
      <FS\_OPTREE>-LEVEL = 3.  
      CLEAR <FS\_OPTREE>-OBJNR.  
      CLEAR <FS\_OPTREE>-SLS\_PRICE\_C.  
      CLEAR <FS\_OPTREE>-SLS\_PRICE.  
      CLEAR <FS\_OPTREE>-PROJ\_POST1.  
      CLEAR <FS\_OPTREE>-PROJ\_PLSEZ.  
      CLEAR <FS\_OPTREE>-POSID.  
      CLEAR <FS\_OPTREE>-PROJ\_STAT.  
      CLEAR <FS\_OPTREE>-TASK\_STAT.  
  
      IF <FS\_OPTREE>-END\_COST IS NOT INITIAL.  
        LV\_ORD\_COST = <FS\_OPTREE>-ORDR\_COST / <FS\_OPTREE>-END\_COST \* 100.  
        WRITE LV\_ORD\_COST TO <FS\_OPTREE>-ORDR\_PERC DECIMALS 2 RIGHT-JUSTIFIED.  
      ELSE.  
        WRITE LV\_ORD\_COST TO <FS\_OPTREE>-ORDR\_PERC DECIMALS 2 RIGHT-JUSTIFIED.  
      ENDIF.  
\*8888888888888 SUBTOTAL  
\*8888888888888 TOTAL  
      APPEND INITIAL LINE TO T\_OPTREE ASSIGNING <FS\_OPTREE>.  
      MOVE-CORRESPONDING LS\_LVL\_ONE TO <FS\_OPTREE>.  
      <FS\_OPTREE>-STEP\_NAME = TEXT-F09.  
      <FS\_OPTREE>-LEVEL = 3.  
      CLEAR <FS\_OPTREE>-OBJNR.  
      CLEAR <FS\_OPTREE>-PROJ\_POST1.  
      CLEAR <FS\_OPTREE>-PROJ\_PLSEZ.  
      CLEAR <FS\_OPTREE>-POSID.  
      CLEAR <FS\_OPTREE>-PROJ\_STAT.  
      CLEAR <FS\_OPTREE>-TASK\_STAT.  
  
      IF <FS\_OPTREE>-END\_COST IS NOT INITIAL.  
        LV\_ORD\_COST = <FS\_OPTREE>-ORDR\_COST / <FS\_OPTREE>-END\_COST \* 100.  
        WRITE LV\_ORD\_COST TO <FS\_OPTREE>-ORDR\_PERC DECIMALS 2 RIGHT-JUSTIFIED.  
      ELSE.  
        WRITE LV\_ORD\_COST TO <FS\_OPTREE>-ORDR\_PERC DECIMALS 2 RIGHT-JUSTIFIED.  
      ENDIF.  
\*8888888888888 TOTAL  
      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>-CURR\_ACTL = LS\_LVL\_ONE-ORIG\_COST - LS\_LVL\_ONE-ORDR\_COST.  
      ENDIF.  
\*8888888888888 LEVEL 0  
      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 .  
          <FS\_OPTREE>-SLS\_PRICE\_C = LS\_LVL\_ONE-SLS\_PRICE\_C.  
        ENDIF.  
      ENDIF.  
\*8888888888888 LEVEL 0  
\*8888888888888 LEVEL 1  
      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.  
        IF <FS\_OPTREE> IS ASSIGNED.  
          LOOP AT LT\_SUBTOT INTO LS\_SUBTOT.  
            LS\_OPTREE = <FS\_OPTREE>.  
            IF SY-TABIX = 1.  
              LS\_OPTREE-SUBTOT = 1.  
            ELSE.  
              LS\_OPTREE-SUBTOT = 2.  
            ENDIF.  
            LS\_OPTREE-ORIG\_COST = LS\_SUBTOT-ORIG\_COST.  
            LS\_OPTREE-CURR\_BUDG = LS\_SUBTOT-CURR\_BUDG.  
            LS\_OPTREE-END\_COST  = LS\_SUBTOT-END\_COST.  
            LS\_OPTREE-ACTL\_COST = LS\_SUBTOT-ACTL\_COST.  
            LS\_OPTREE-ORDR\_COST = LS\_SUBTOT-ORDR\_COST.  
            LS\_OPTREE-CURR\_ACTL = LS\_SUBTOT-CURR\_ACTL.  
            LS\_OPTREE-SLS\_PRICE = LS\_SUBTOT-SLS\_PRICE .  
            CLEAR LS\_OPTREE-PROJ\_POST1.  
            CLEAR LS\_OPTREE-PROJ\_PLSEZ.  
            CLEAR LS\_OPTREE-POSID.  
            CLEAR LS\_OPTREE-OBJNR.  
            CLEAR LS\_OPTREE-PROJ\_STAT.  
            CLEAR LS\_OPTREE-TASK\_STAT.  
  
            IF LS\_SUBTOT-END\_COST IS NOT INITIAL.  
              LV\_ORD\_COST = LS\_SUBTOT-ORDR\_COST / LS\_SUBTOT-END\_COST \* 100.  
              WRITE LV\_ORD\_COST TO LS\_OPTREE-ORDR\_PERC DECIMALS 2 RIGHT-JUSTIFIED.  
            ELSE.  
              WRITE LV\_ORD\_COST TO LS\_OPTREE-ORDR\_PERC DECIMALS 2 RIGHT-JUSTIFIED.  
            ENDIF.  
            CONCATENATE  LS\_OPTREE-ORDR\_PERC C\_% INTO LS\_OPTREE-ORDR\_PERC.  
            CASE LS\_SUBTOT-DEPT .  
              WHEN C\_E.  
                LS\_OPTREE-STEP\_NAME = TEXT-F01.  
              WHEN C\_M.  
                LS\_OPTREE-STEP\_NAME  = TEXT-F02.  
              WHEN C\_I.  
                LS\_OPTREE-STEP\_NAME  = TEXT-F03.  
              WHEN C\_J.  
                LS\_OPTREE-STEP\_NAME  = TEXT-F04.  
              WHEN C\_S.  
                LS\_OPTREE-STEP\_NAME  = TEXT-F05.  
              WHEN OTHERS.  
                LS\_OPTREE-STEP\_NAME  = TEXT-F06.  
            ENDCASE.  
            APPEND  LS\_OPTREE TO T\_OPTREE.  
            CLEAR LV\_ORD\_COST.  
            CLEAR LS\_OPTREE.  
          ENDLOOP.  
        ENDIF.  
      ENDIF.  
\*8888888888888 LEVEL 1  
    ENDLOOP.  
    DESCRIBE TABLE T\_OUTPUT LINES V\_RECORDS.  
  ENDMETHOD.                    "FORMAT\_DATA  
  METHOD REG\_EVENTS.  
    R\_EVENTS = R\_TREE->GET\_EVENT( ).  
  
    SET HANDLER ME->ON\_DOUBLE\_CLICK FOR R\_EVENTS.  
    SET HANDLER ME->ON\_LINK\_CLICK FOR R\_EVENTS.  
    SET HANDLER ME->ON\_BUTTON\_CLICK FOR R\_EVENTS.  
  ENDMETHOD.                    "REG\_EVENTS  
  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\_OPTREE 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.                    "SUPPLY\_DAT  
  METHOD SAVE\_FILE.  
    CHECK ERROR\_CHECK( ) <> ABAP\_TRUE.  
    IF DWNLBT IS NOT INITIAL.  
      SAVE\_BFILE( ).  
    ELSEIF DWNLFF IS NOT INITIAL.  
      SAVE\_FFILE( ).  
    ENDIF.  
  ENDMETHOD.  
  METHOD SAVE\_BFILE.  
    DATA: LV\_FNAME  TYPE STRING,  
          LV\_LEN TYPE I VALUE 0,  
          LS\_RESULT TYPE SOLISTI1.  
  
  
    CHECK CHECK\_BFILE( EXPORTING BTPATH = BTPATH  
                 IMPORTING FILE = LV\_FNAME ) IS INITIAL.  
    CHECK LV\_FNAME IS NOT INITIAL.  
    CHECK CREATE\_CSV\_OP( ) <> ABAP\_TRUE.  
\*    CONCATENATE LV\_FNAME TEXT-HD3 SY-DATUM SY-UZEIT C\_CSV INTO LV\_FNAME.  
    OPEN DATASET LV\_FNAME FOR INPUT IN TEXT MODE ENCODING DEFAULT.  
    IF SY-SUBRC = 0.  
      LOOP AT T\_RESULT INTO LS\_RESULT.  
        LV\_LEN = STRLEN( LS\_RESULT ).  
        TRANSFER LS\_RESULT TO LV\_FNAME LENGTH LV\_LEN.  
      ENDLOOP.  
      CLOSE DATASET LV\_FNAME.  
      IF SY-SUBRC <> 0.  
        V\_ERROR\_CHECK = ABAP\_TRUE.  
      ELSE.  
        V\_ERROR\_CHECK = ABAP\_FALSE.  
      ENDIF.  
    ELSE.  
      V\_ERROR\_CHECK = ABAP\_TRUE.  
    ENDIF.  
    IF V\_ERROR\_CHECK = ABAP\_TRUE.  
      MESSAGE  W000(KN) WITH LV\_FNAME ' File download failed '  .  
    ELSE.  
      MESSAGE S000(KN) WITH LV\_FNAME ' File downloaded successfully'  .  
    ENDIF.  
  ENDMETHOD.  
  METHOD SAVE\_FFILE.  
    DATA: LV\_FNAME         TYPE STRING.  
  
    CHECK CREATE\_CSV\_OP( ) <> ABAP\_TRUE.  
    LV\_FNAME = 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                  = T\_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.  
        MESSAGE  S000(KN) WITH 'File ' LV\_FNAME ' downloaded successfully' .  
      ENDIF.  
    ELSE.  
      MESSAGE  S000(KN) WITH 'File not found'.  
    ENDIF.  
  
  ENDMETHOD.                    "SAVE\_FFILE  
  METHOD SET\_BPATH.  
  
    DATA: LR\_CONTENT TYPE REF TO OBJECT,  
          LV\_VALUE TYPE ZVALUE.  
  
    TRY .  
        CREATE OBJECT LR\_CONTENT TYPE ('ZDAC\_CL\_CUSTOMER\_CONFIG')  
          EXPORTING  
            \_CODE = 'PRJ\_CST\_SUM\_BATCH'.  
        CALL METHOD lr\_content->('GET\_VALUE')  
          IMPORTING  
            \_VALUE = LV\_VALUE.  
        BTPATH = LV\_VALUE.  
      CATCH CX\_ROOT.  
        CLEAR BTPATH.  
    ENDTRY.  
  
  
  
  ENDMETHOD.  
  METHOD SCR\_CHECK.  
  
    DATA: V\_VBUKR      TYPE PS\_VBUKR,  
          V\_VKOKR      TYPE PS\_VKOKR,  
          V\_STAT       TYPE J\_STATUS,  
          T\_PSPID      TYPE TABLE OF PS\_PSPID,  
          T\_POSID      TYPE TABLE OF PS\_POSID,  
          T\_POSKI      TYPE TABLE OF PS\_POSKI,  
          T\_PRCTR      TYPE TABLE OF PRCTR,  
          T\_PRART      TYPE TABLE OF PS\_PRART.  
  
    FIELD-SYMBOLS:  
      <R\_SELECT> TYPE RSDSSELOPT,  
      <T\_POSID> LIKE LINE OF T\_POSID,  
      <T\_POSKI> LIKE LINE OF T\_POSKI,  
      <T\_PSPID> LIKE LINE OF T\_PSPID,  
      <T\_PRCTR> LIKE LINE OF T\_PRCTR,  
      <T\_PRART> LIKE LINE OF T\_PRART.  
  
\*\*\*Must have input in at least one  
    IF PSPID[] IS INITIAL AND  
       POSID[] IS INITIAL AND  
       POSKI[] IS INITIAL.  
      RVAL = ABAP\_TRUE.  
      MESSAGE E000(KN) WITH 'Enter atleast Project Definition or WBS '.  
      EXIT.  
    ELSE.  
\*\*\*check company code  
      IF VBUKR IS NOT INITIAL.  
        SELECT BUKRS INTO V\_VBUKR  
          FROM  T001  
          WHERE BUKRS = VBUKR.  
        ENDSELECT.  
        IF SY-SUBRC <> 0.  
          RVAL = ABAP\_TRUE.  
          MESSAGE E000(KN) WITH 'Invalid Company Code: ' VBUKR.  
        ENDIF.  
      ENDIF.  
  
  
\*\*\*Check Controlling Area  
      IF VKOKR IS NOT INITIAL.  
        SELECT KOKRS INTO V\_VKOKR  
          FROM  TKA01  
          WHERE KOKRS = VKOKR.  
        ENDSELECT.  
        IF SY-SUBRC <> 0.  
          RVAL = ABAP\_TRUE.  
          MESSAGE E000(KN) WITH 'Invalid Controlling Area: ' VKOKR.  
        ENDIF.  
      ENDIF.  
  
  
\*\*\*Check STATUS  
      IF STAT IS NOT INITIAL.  
        SELECT STAT INTO V\_STAT  
          FROM  JEST  
          WHERE STAT = STAT.  
        ENDSELECT.  
        IF SY-SUBRC <> 0.  
          RVAL = ABAP\_TRUE.  
          MESSAGE E000(KN) WITH 'Invalid status: ' STAT.  
        ENDIF.  
      ENDIF.  
  
  
\*\*\* Check Project Definition  
      SELECT PSPID INTO TABLE T\_PSPID  
        FROM PROJ  
        WHERE PSPID IN PSPID.  
      SORT T\_PSPID BY TABLE\_LINE ASCENDING.  
  
      IF T\_PSPID IS INITIAL.  
        LOOP AT PSPID ASSIGNING <R\_SELECT> WHERE SIGN <> C\_E.  
          READ TABLE T\_PSPID ASSIGNING <T\_PSPID>  
            WITH KEY TABLE\_LINE = <R\_SELECT>-LOW  
            BINARY SEARCH.  
          IF SY-SUBRC <> 0.  
            RVAL = ABAP\_TRUE.  
            MESSAGE E000(KN) WITH 'Invalid Project: ' <R\_SELECT>-LOW.  
          ENDIF.  
        ENDLOOP.  
      ENDIF.  
  
  
\*\*\* Check WBS Element  
      SELECT POSID INTO TABLE T\_POSID  
        FROM PRPS  
        WHERE POSID IN POSID.  
      SORT T\_POSID BY TABLE\_LINE ASCENDING.  
  
      IF T\_POSID IS INITIAL.  
        LOOP AT POSID ASSIGNING <R\_SELECT> WHERE SIGN <> C\_E.  
          READ TABLE T\_POSID ASSIGNING <T\_POSID>  
            WITH KEY TABLE\_LINE = <R\_SELECT>-LOW  
            BINARY SEARCH.  
          IF SY-SUBRC <> 0.  
            RVAL = ABAP\_TRUE.  
            MESSAGE E000(KN) WITH 'Invalid WBS Element ' <R\_SELECT>-LOW.  
          ENDIF.  
        ENDLOOP.  
      ENDIF.  
  
  
\*\*\* Check WBS Short ID  
      SELECT POSKI INTO TABLE T\_POSKI  
        FROM PRPS  
        WHERE POSKI IN POSKI.  
      SORT T\_POSKI BY TABLE\_LINE ASCENDING.  
  
      IF T\_POSKI IS INITIAL.  
        LOOP AT POSKI ASSIGNING <R\_SELECT> WHERE SIGN <> C\_E.  
          READ TABLE T\_POSKI ASSIGNING <T\_POSKI>  
            WITH KEY TABLE\_LINE = <R\_SELECT>-LOW  
            BINARY SEARCH.  
          IF SY-SUBRC <> 0.  
            RVAL = ABAP\_TRUE.  
            MESSAGE E000(KN) WITH 'Invalid WBS ShortId: ' <R\_SELECT>-LOW.  
          ENDIF.  
        ENDLOOP.  
      ENDIF.  
  
  
\*\*\* Check Profit Center  
      SELECT PRCTR INTO TABLE T\_PRCTR  
        FROM CEPC BYPASSING BUFFER  
        WHERE PRCTR IN PRCTR .  
      SORT T\_PRCTR BY TABLE\_LINE ASCENDING.  
  
      IF T\_PRCTR IS INITIAL.  
        LOOP AT PRCTR ASSIGNING <R\_SELECT> WHERE SIGN <> C\_E.  
          READ TABLE T\_PRCTR ASSIGNING <T\_PRCTR>  
            WITH KEY TABLE\_LINE = <R\_SELECT>-LOW  
            BINARY SEARCH.  
          IF SY-SUBRC <> 0.  
            RVAL = ABAP\_TRUE.  
            MESSAGE E000(KN) WITH 'Invalid Profit Center: ' <R\_SELECT>-LOW.  
          ENDIF.  
        ENDLOOP.  
      ENDIF.  
  
  
\*\*\* Check Project Type  
      SELECT PRART INTO TABLE T\_PRART  
        FROM TCJ1  
        WHERE PRART IN PRART.  
      SORT T\_PRART BY TABLE\_LINE ASCENDING.  
  
      IF T\_PRART IS INITIAL.  
        LOOP AT PRART ASSIGNING <R\_SELECT> WHERE SIGN <> C\_E.  
          READ TABLE T\_PRART ASSIGNING <T\_PRART>  
            WITH KEY TABLE\_LINE = <R\_SELECT>-LOW  
            BINARY SEARCH.  
          IF SY-SUBRC <> 0.  
            RVAL = ABAP\_TRUE.  
            MESSAGE I000(KN) WITH 'Invalid Project Type: ' <R\_SELECT>-LOW.  
          ENDIF.  
        ENDLOOP.  
      ENDIF.  
    ENDIF.  
  ENDMETHOD.                    "SCREEN\_FIELD\_CHECK  
  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 .  
      LS\_RETURN-TYPE = C\_E.  
      LS\_RETURN-MESSAGE = C\_NO\_DATA.  
      APPEND LS\_RETURN TO T\_LOG.  
    ENDIF.  
  
  ENDMETHOD.                    "WRITE\_LOG  
  
ENDCLASS.                    "LCL\_CNTLR IMPLEMENTATION  
  
DATA: OK\_CODE TYPE SYUCOMM,  
      GS\_RSDSSELOPT TYPE RSDSSELOPT,  
      GT\_PSPID TYPE RSELOPTION,  
      GT\_POSID TYPE RSELOPTION,  
      GT\_POSKI TYPE RSELOPTION,  
      GT\_PRCTR TYPE RSELOPTION,  
      GT\_PRART TYPE RSELOPTION.  
  
DATA: GR\_CL\_CONTROLLER TYPE REF TO LCL\_CNTLR.  
\*----------------------------------------------------------------------\*  
\* SELECTION-SCREEN  
\*----------------------------------------------------------------------\*  
  
SELECTION-SCREEN BEGIN OF BLOCK SELECTION WITH FRAME TITLE TEXT-SL1.  
PARAMETERS:  
P\_VBUKR TYPE PROJ-VBUKR OBLIGATORY DEFAULT '1000' VALUE CHECK,  
P\_VKOKR TYPE PROJ-VKOKR OBLIGATORY DEFAULT '1000' VALUE CHECK,  
P\_STAT  TYPE JEST-STAT NO-DISPLAY,  
P\_PROF  TYPE TCNT-PROF\_DB DEFAULT '000000000001'  
                          NO-DISPLAY.  
  
SELECT-OPTIONS:  
S\_PSPID FOR PROJ-PSPID,  
S\_POSID FOR PRPS-POSID,  
S\_POSKI FOR PRPS-POSKI,  
S\_PRCTR FOR PRPS-PRCTR,  
S\_PRART FOR PRPS-PRART.  
  
PARAMETER P\_REPORT  RADIOBUTTON GROUP RB1 DEFAULT 'X' USER-COMMAND ABC.  
SELECTION-SCREEN BEGIN OF LINE.  
PARAMETER P\_DWNFF RADIOBUTTON GROUP RB1 MODIF ID MD1 .  
SELECTION-SCREEN: COMMENT 3(30) TEXT-SC1,  
                  POSITION 33.  
PARAMETER P\_EXPATH  TYPE RLGRAP-FILENAME MODIF ID MD2.  
SELECTION-SCREEN END OF LINE .  
SELECTION-SCREEN BEGIN OF LINE .  
PARAMETER P\_DWNBT   RADIOBUTTON GROUP RB1 MODIF ID MD1.  
SELECTION-SCREEN: COMMENT 3(30) TEXT-SC0,  
                  POSITION 33.  
PARAMETER P\_BTPATH  TYPE RLGRAP-FILENAME MODIF ID MD3.  
SELECTION-SCREEN END OF LINE .  
SELECTION-SCREEN END OF BLOCK SELECTION.  
  
INITIALIZATION.  
  LCL\_CNTLR=>SET\_BPATH(  
    IMPORTING  
      BTPATH = P\_BTPATH  
  ).  
  
AT SELECTION-SCREEN OUTPUT.  
  IF P\_REPORT IS NOT INITIAL.  
    LOOP AT SCREEN.  
      CASE SCREEN-GROUP1 .  
        WHEN 'MD2' OR 'MD3'.  
          SCREEN-ACTIVE = 0.  
          MODIFY SCREEN.  
      ENDCASE.  
    ENDLOOP.  
  ELSEIF  P\_DWNFF IS NOT INITIAL.  
    LOOP AT SCREEN.  
      IF SCREEN-GROUP1 EQ 'MD2'.  
        SCREEN-ACTIVE = 1.  
        MODIFY SCREEN.  
      ELSEIF SCREEN-GROUP1 EQ 'MD3'.  
        SCREEN-ACTIVE = 0.  
        MODIFY SCREEN.  
      ENDIF.  
    ENDLOOP.  
  ELSEIF P\_DWNBT IS NOT INITIAL.  
    LOOP AT SCREEN .  
      IF SCREEN-GROUP1 EQ 'MD3'.  
        SCREEN-ACTIVE = 1.  
        MODIFY SCREEN.  
      ELSEIF SCREEN-GROUP1 EQ 'MD2'.  
        SCREEN-ACTIVE = 0.  
        MODIFY SCREEN.  
      ENDIF.  
    ENDLOOP.  
  ENDIF.  
  
AT SELECTION-SCREEN.  
  LOOP AT S\_PSPID.  
    MOVE-CORRESPONDING S\_PSPID TO GS\_RSDSSELOPT.  
    APPEND GS\_RSDSSELOPT TO GT\_PSPID.  
  ENDLOOP.  
  LOOP AT S\_POSID.  
    MOVE-CORRESPONDING S\_POSID TO GS\_RSDSSELOPT.  
    APPEND GS\_RSDSSELOPT TO GT\_POSID.  
  ENDLOOP.  
  LOOP AT S\_POSKI.  
    MOVE-CORRESPONDING S\_POSKI TO GS\_RSDSSELOPT.  
    APPEND GS\_RSDSSELOPT TO GT\_POSKI.  
  ENDLOOP.  
  LOOP AT S\_PRCTR.  
    MOVE-CORRESPONDING S\_PRCTR TO GS\_RSDSSELOPT.  
    APPEND GS\_RSDSSELOPT TO GT\_PRCTR.  
  ENDLOOP.  
  LOOP AT S\_PRART.  
    MOVE-CORRESPONDING S\_PRART TO GS\_RSDSSELOPT.  
    APPEND GS\_RSDSSELOPT TO GT\_PRART.  
  ENDLOOP.  
  CHECK LCL\_CNTLR=>SCR\_CHECK(  
        VBUKR  = P\_VBUKR  
        VKOKR  = P\_VKOKR  
        STAT   = P\_STAT  
        EXPATH = P\_EXPATH  
        PSPID  = GT\_PSPID  
        POSID  = GT\_POSID  
        POSKI  = GT\_POSKI  
        PRCTR  = GT\_PRCTR  
        PRART  = GT\_PRART  
   ) <> ABAP\_TRUE.  
  CHECK LCL\_CNTLR=>CHECK\_FFILE(  
    EXPORTING EXPATH = P\_EXPATH )  
  <> ABAP\_TRUE.  
  CHECK LCL\_CNTLR=>CHECK\_BFILE(  
    EXPORTING BTPATH = P\_BTPATH )  
  <> ABAP\_TRUE.  
  
  
AT SELECTION-SCREEN ON VALUE-REQUEST FOR P\_EXPATH.  
  LCL\_CNTLR=>F4\_HLPFILE( CHANGING EXPATH = P\_EXPATH ).  
  
START-OF-SELECTION.  
  CREATE OBJECT GR\_CL\_CONTROLLER  
    EXPORTING  
      VBUKR   = P\_VBUKR  
      VKOKR   = P\_VKOKR  
      STAT    = P\_STAT  
      PROF\_DB = P\_PROF  
      DWNLFF  = P\_DWNFF  
      DWNLBT  = P\_DWNBT  
      EXPATH  = P\_EXPATH  
      BTPATH  = P\_BTPATH  
      PSPID   = GT\_PSPID  
      POSID   = GT\_POSID  
      POSKI   = GT\_POSKI  
      PRCTR   = GT\_PRCTR  
      PRART   = GT\_PRART.  
  
END-OF-SELECTION.  
  CHECK GR\_CL\_CONTROLLER IS BOUND.  
  GR\_CL\_CONTROLLER->INIT\_ALV( ).  
  GR\_CL\_CONTROLLER->SAVE\_FILE( ).