\*&---------------------------------------------------------------------\*  
\*& Report  ZBDC\_CREATE\_PO\_01  
\*&  
\*&---------------------------------------------------------------------\*  
\*&  
\*&  
\*&---------------------------------------------------------------------\*  
  
REPORT  zbdc\_create\_po\_01.  
INCLUDE zbdc\_create\_po\_01\_top.  
  
AT SELECTION-SCREEN ON VALUE-REQUEST FOR p\_path.  
  PERFORM get\_path.  
  
START-OF-SELECTION.  
  PERFORM upload\_data.  
  PERFORM validate\_and\_convert\_data.  
  PERFORM create\_po.  
  
END-OF-SELECTION.  
  PERFORM display\_log.  
  
  INCLUDE zbdc\_create\_po\_01\_forms.

\*&---------------------------------------------------------------------\*  
\*&  Include           ZBDC\_CREATE\_PO\_01\_TOP  
\*&---------------------------------------------------------------------\*  
TYPES:  
 BEGIN OF t\_data,  
   lifnr(10)  TYPE c,  
   bsart(4)   TYPE c,  
   bedat(10)  TYPE c,  
   ekorg(4)   TYPE c,  
   ekgrp(3)   TYPE n,  
   matnr(18)  TYPE c,  
   menge(17)  TYPE c,  
   meins(3)   TYPE c,  
   eeind(10)  TYPE c,  
   werks(4)   TYPE c,  
   lgort(4)   TYPE c,  
 END OF t\_data,  
 t\_data\_tab TYPE STANDARD TABLE OF t\_data,  
  
  BEGIN OF t\_po\_head,  
    po     TYPE i,  
    lifnr TYPE ekko-lifnr,  
    bsart TYPE rm06e-bsart,  
\*    bedat TYPE rm06e-bedat,  
    bedat TYPE c LENGTH 10,  
    ekorg TYPE ekko-ekorg,  
    ekgrp TYPE ekko-ekgrp,  
 END OF t\_po\_head,  
 t\_po\_head\_tab TYPE STANDARD TABLE OF t\_po\_head,  
  
 BEGIN OF t\_po\_item,  
   po     TYPE i,  
   matnr  TYPE ekpo-ematn,  
\*   menge  TYPE ekpo-menge,  
   menge(6)  TYPE c,  
   meins(3)  TYPE c,  
   eeind  TYPE rm06e-eeind,  
   werks  TYPE ekpo-werks,  
\*   lgort  TYPE ekpo-lgort,  
   lgort(4)  TYPE n,  
 END OF t\_po\_item,  
 t\_po\_item\_tab TYPE STANDARD TABLE OF t\_po\_item,  
 BEGIN OF t\_log,  
   index TYPE sy-tabix,  
   msg TYPE string,  
   END OF t\_log,  
   t\_log\_tab TYPE STANDARD TABLE OF t\_log.  
  
  
DATA: it\_data       TYPE t\_data\_tab,  
      it\_po\_head    TYPE t\_po\_head\_tab,  
      it\_po\_item    TYPE t\_po\_item\_tab,  
      it\_log        TYPE t\_log\_tab,  
      it\_bdcdata    TYPE STANDARD TABLE OF bdcdata,  
      it\_msgcoll    TYPE STANDARD TABLE OF bdcmsgcoll,  
      wa\_data       TYPE t\_data,  
      wa\_po\_head    TYPE t\_po\_head,  
      wa\_po\_item    TYPE t\_po\_item,  
      wa\_log        TYPE t\_log,  
      wa\_bdcdata    TYPE bdcdata,  
      wa\_msgcoll    TYPE bdcmsgcoll.  
  
DATA: g\_index TYPE sy-tabix.  
  
PARAMETERS: p\_path TYPE ibipparms-path.  
PARAMETERS: p\_mode TYPE ctu\_mode.

\*&---------------------------------------------------------------------\*  
\*&  Include           ZBDC\_CREATE\_PO\_01\_FORMS  
\*&---------------------------------------------------------------------\*  
\*&---------------------------------------------------------------------\*  
\*&      Form  GET\_PATH  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*  
\*  -->  p1        text  
\*  <--  p2        text  
\*----------------------------------------------------------------------\*  
FORM get\_path .  
  CALL FUNCTION 'F4\_FILENAME'  
    IMPORTING  
      file\_name = p\_path.  
ENDFORM.                    " GET\_PATH  
\*&---------------------------------------------------------------------\*  
\*&      Form  UPLOAD\_DATA  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*  
\*  -->  p1        text  
\*  <--  p2        text  
\*----------------------------------------------------------------------\*  
FORM upload\_data .  
  DATA:  
        l\_file TYPE rlgrap-filename,  
        l\_data TYPE truxs\_t\_text\_data.  
  MOVE p\_path TO l\_file.  
  CALL FUNCTION 'TEXT\_CONVERT\_XLS\_TO\_SAP'  
    EXPORTING  
      i\_field\_seperator    = 'X'  
      i\_line\_header        = 'X'  
      i\_tab\_raw\_data       = l\_data  
      i\_filename           = l\_file  
    TABLES  
      i\_tab\_converted\_data = it\_data  
    EXCEPTIONS  
      conversion\_failed    = 1  
      OTHERS               = 2.  
  IF sy-subrc <> 0.  
\* Implement suitable error handling here  
  ENDIF.  
ENDFORM.                    " UPLOAD\_DATA  
\*&---------------------------------------------------------------------\*  
\*&      Form  VALIDATE\_AND\_CONVERT\_DATA  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*  
\*  -->  p1        text  
\*  <--  p2        text  
\*----------------------------------------------------------------------\*  
FORM validate\_and\_convert\_data .  
  DATA: l\_index     TYPE i.  
  LOOP AT it\_data INTO wa\_data.  
    IF wa\_data-lifnr IS NOT INITIAL.  
      "Collect  Header Details  
      l\_index = l\_index + 1.  
      wa\_po\_head-po = l\_index.  
      "Document Type  
      wa\_po\_head-bsart = wa\_data-bsart.  
      "Vendor  
      CALL FUNCTION 'CONVERSION\_EXIT\_ALPHA\_OUTPUT'  
        EXPORTING  
          input  = wa\_data-lifnr  
        IMPORTING  
          output = wa\_po\_head-lifnr.  
      "Document Date  
      wa\_po\_head-bedat = wa\_data-bedat.  
      wa\_po\_head-ekorg = wa\_data-ekorg.  
      wa\_po\_head-ekgrp = wa\_data-ekgrp.  
      APPEND wa\_po\_head TO it\_po\_head.  
    ENDIF.  
    "Collect Item Details  
    wa\_po\_item-po     = l\_index.  
    "Material Number  
    CALL FUNCTION 'CONVERSION\_EXIT\_MATN1\_OUTPUT'  
      EXPORTING  
        input  = wa\_data-matnr  
      IMPORTING  
        output = wa\_po\_item-matnr.  
    "Quantity  
    wa\_po\_item-menge  = wa\_data-menge.  
    "Unit of Measurement  
    CALL FUNCTION 'CONVERSION\_EXIT\_CUNIT\_OUTPUT'  
      EXPORTING  
        input          = wa\_data-meins  
        language       = sy-langu  
      IMPORTING  
\*       LONG\_TEXT      =  
        output         = wa\_po\_item-meins  
\*       SHORT\_TEXT     =  
      EXCEPTIONS  
        unit\_not\_found = 1  
        OTHERS         = 2.  
    IF sy-subrc <> 0.  
\* Implement suitable error handling here  
    ENDIF.  
    "Item Delivery Date  
    wa\_po\_item-eeind = wa\_data-eeind.  
    "Plant  
    wa\_po\_item-werks = wa\_data-werks.  
    "Storage Location  
    wa\_po\_item-lgort = wa\_data-lgort.  
    APPEND wa\_po\_item TO it\_po\_item.  
  ENDLOOP.  
  SORT:  
   it\_po\_head BY po,  
   it\_po\_item BY po.  
ENDFORM.                    " VALIDATE\_AND\_CONVERT\_DATA  
\*&---------------------------------------------------------------------\*  
\*&      Form  CREATE\_PO  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*  
\*  -->  p1        text  
\*  <--  p2        text  
\*----------------------------------------------------------------------\*  
FORM create\_po.  
  DATA:  
        l\_nindex(2) TYPE n,  
        l\_field(30) TYPE c,  
        l\_quantity(5) TYPE c.  
  LOOP AT it\_po\_head INTO wa\_po\_head.  
    g\_index = sy-tabix.  
    "Collect Header data  
    PERFORM bdc\_dynpro      USING 'SAPMM06E' '0100'.  
    PERFORM bdc\_field       USING 'BDC\_OKCODE'  
                            '/00'.  
    PERFORM bdc\_field       USING 'EKKO-LIFNR'  
                                  wa\_po\_head-lifnr.  
    PERFORM bdc\_field       USING 'RM06E-BSART'  
                                  wa\_po\_head-bsart.  
    PERFORM bdc\_field       USING 'RM06E-BEDAT'  
                                  wa\_po\_head-bedat.  
    PERFORM bdc\_field       USING 'EKKO-EKORG'  
                                  wa\_po\_head-ekorg.  
    PERFORM bdc\_field       USING 'EKKO-EKGRP'  
                                  wa\_po\_head-ekgrp.  
    PERFORM bdc\_dynpro      USING 'SAPMM06E' '0120'.  
    PERFORM bdc\_field       USING 'BDC\_OKCODE'  
                              'BU'.  
    READ TABLE it\_po\_item TRANSPORTING NO FIELDS  
                          WITH KEY po = wa\_po\_head-po.  
    IF sy-subrc NE 0.  
      CONTINUE.  
    ENDIF.  
    LOOP AT it\_po\_item  INTO wa\_po\_item  
                        FROM sy-tabix.  
      IF wa\_po\_head-po NE wa\_po\_item-po.  
        EXIT.  
      ENDIF.  
      "Collect Item data  
      MOVE sy-tabix TO l\_nindex.  
      "Material  
      CLEAR l\_field.  
      PERFORM build\_field USING 'EKPO-EMATN' l\_nindex  
                          CHANGING l\_field.  
      PERFORM bdc\_field       USING l\_field  
                                wa\_po\_item-matnr.  
      "Quantity  
      CLEAR l\_field.  
      l\_quantity = wa\_po\_item-menge.  
      PERFORM build\_field USING 'EKPO-MENGE' l\_nindex  
                          CHANGING l\_field.  
      PERFORM bdc\_field       USING l\_field  
                                wa\_po\_item-menge.  
      "Unit of Measurement  
      CLEAR l\_field.  
      PERFORM build\_field USING 'EKPO-MEINS' l\_nindex  
                          CHANGING l\_field.  
      PERFORM bdc\_field       USING l\_field  
                                    wa\_po\_item-meins.  
      "Delivery Date  
      CLEAR l\_field.  
      PERFORM build\_field USING 'RM06E-EEIND' l\_nindex  
                          CHANGING l\_field.  
      PERFORM bdc\_field       USING l\_field  
                                wa\_po\_item-eeind.  
      "Plant  
      CLEAR l\_field.  
      PERFORM build\_field USING 'EKPO-WERKS' l\_nindex  
                          CHANGING l\_field.  
      PERFORM bdc\_field       USING l\_field  
                                wa\_po\_item-werks.  
      "Storage Location  
      CLEAR l\_field.  
      PERFORM build\_field USING 'EKPO-LGORT' l\_nindex  
                          CHANGING l\_field.  
      PERFORM bdc\_field       USING l\_field  
                                wa\_po\_item-lgort.  
    ENDLOOP.  
    "Call Transaction  
    CALL TRANSACTION          'ME21'  
              USING           it\_bdcdata  
               MODE           p\_mode  
               UPDATE         'S'  
               MESSAGES INTO  it\_msgcoll.  
    PERFORM collect\_msg.  
    CLEAR: it\_bdcdata, it\_msgcoll.  
  ENDLOOP.  
ENDFORM.                    " CREATE\_PO  
\*&---------------------------------------------------------------------\*  
\*&      Form  DISPLAY\_LOG  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*  
\*  -->  p1        text  
\*  <--  p2        text  
\*----------------------------------------------------------------------\*  
FORM display\_log .  
  SORT it\_log BY index.  
  LOOP AT it\_log INTO wa\_log.  
    AT NEW index.  
      WRITE: / 'Record', wa\_log-index COLOR 5.  
    ENDAT.  
    WRITE: /5 wa\_log-msg.  
  ENDLOOP.  
ENDFORM.                    " DISPLAY\_LOG  
\*----------------------------------------------------------------------\*  
\*        Start new screen                                              \*  
\*----------------------------------------------------------------------\*  
FORM bdc\_dynpro USING program dynpro.  
  CLEAR wa\_bdcdata.  
  wa\_bdcdata-program  = program.  
  wa\_bdcdata-dynpro   = dynpro.  
  wa\_bdcdata-dynbegin = 'X'.  
  APPEND wa\_bdcdata TO it\_bdcdata.  
ENDFORM.                    "BDC\_DYNPRO  
  
\*----------------------------------------------------------------------\*  
\*        Insert field                                                  \*  
\*----------------------------------------------------------------------\*  
FORM bdc\_field USING fnam fval.  
  CLEAR wa\_bdcdata.  
  wa\_bdcdata-fnam = fnam.  
  wa\_bdcdata-fval = fval.  
  APPEND wa\_bdcdata TO it\_bdcdata.  
ENDFORM.                    "BDC\_FIELD  
\*&---------------------------------------------------------------------\*  
\*&      Form  COLLECT\_MSG  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*  
\*  -->  p1        text  
\*  <--  p2        text  
\*----------------------------------------------------------------------\*  
FORM collect\_msg .  
  CLEAR wa\_log.  
  wa\_log-index = g\_index.  
  LOOP AT it\_msgcoll INTO wa\_msgcoll.  
    CALL FUNCTION 'FORMAT\_MESSAGE'  
      EXPORTING  
        id        = wa\_msgcoll-msgid  
        lang      = wa\_msgcoll-msgspra  
        no        = wa\_msgcoll-msgnr  
        v1        = wa\_msgcoll-msgv1  
        v2        = wa\_msgcoll-msgv2  
        v3        = wa\_msgcoll-msgv3  
        v4        = wa\_msgcoll-msgv4  
      IMPORTING  
        msg       = wa\_log-msg  
      EXCEPTIONS  
        not\_found = 1  
        OTHERS    = 2.  
    IF sy-subrc <> 0.  
\* Implement suitable error handling here  
    ENDIF.  
    APPEND wa\_log TO it\_log.  
  ENDLOOP.  
ENDFORM.                    " COLLECT\_MSG  
\*&---------------------------------------------------------------------\*  
\*&      Form  BUILD\_FIELD  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*  
\*      -->P\_0324   text  
\*      -->P\_L\_CINDEX  text  
\*      <--P\_L\_FIELD  text  
\*----------------------------------------------------------------------\*  
FORM build\_field  USING p\_field  
                        p\_index  
                  CHANGING p\_output.  
  CONCATENATE p\_field  
              '('  
              p\_index  
               ')'  
          INTO p\_output.  
ENDFORM.                    " BUILD\_FIELD