CASE 3: Conversion (UPLOAD) Creation

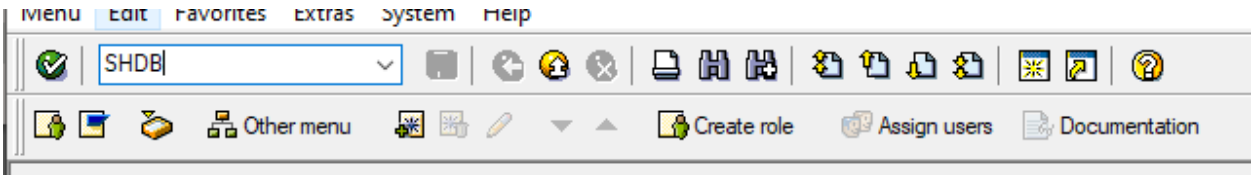
Requirement

Create an ABAP program that will let you upload and create Domains from an excel file.

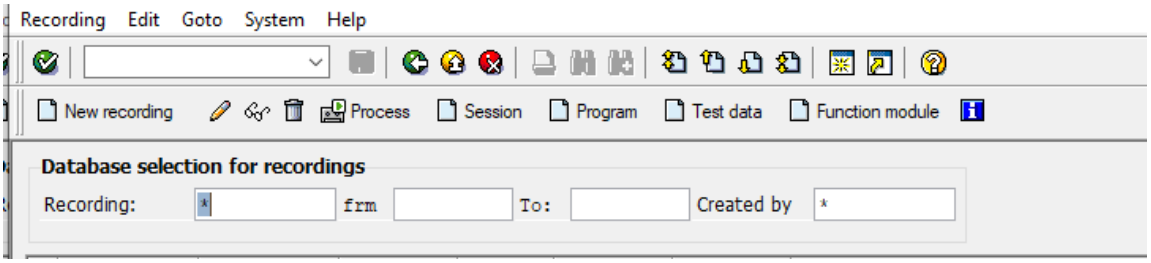
Process

1. Create an ABAP Program to read an excel file.
2. The ABAP program will mass create Domain in SE11 and save it a local package ($TMP)
3. Fields in the excel must contain the following:
    a. Domain Name
    b. Description
    c. Data Type
    d. No. of Characters
    e. Decimal Places
4. You may record using the SHDB transaction for the BDC (Batch Input Session)
5. In each loop in the records show a log after. You can use WRITE commands or show the report in other output methods.

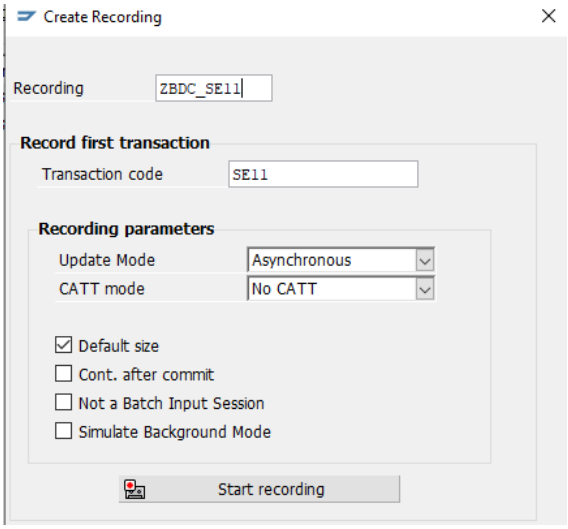
To Start BDC, go to SHDB for recording



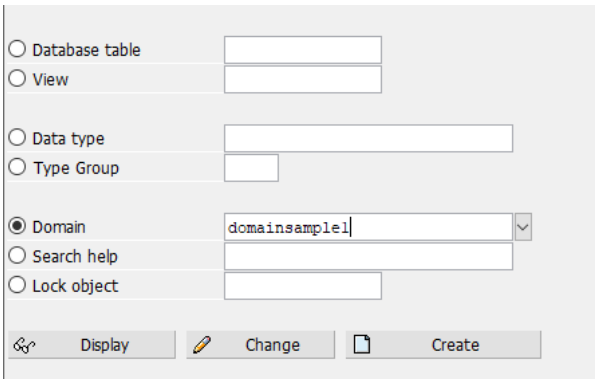
Click on Recording



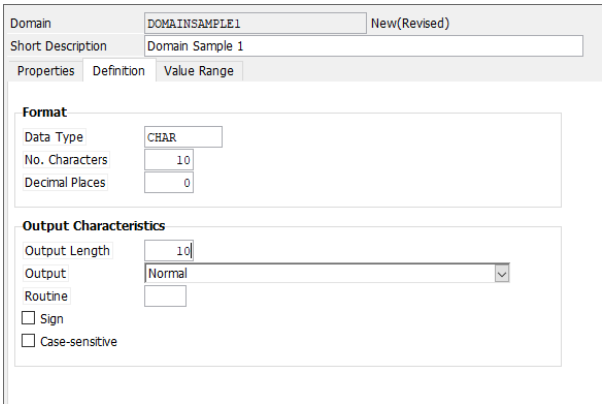
Put the recording name and the transaction code and hit start recording



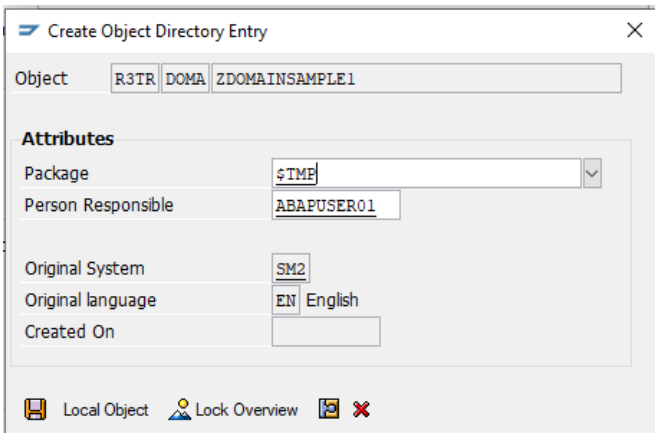
Record how you declare Domain in SE11. Put a name and hit Create



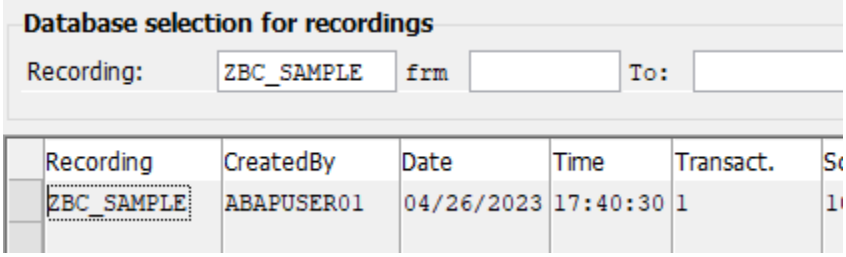
Put a short description, Data type, No. Characters, Decimal Places, and output length.



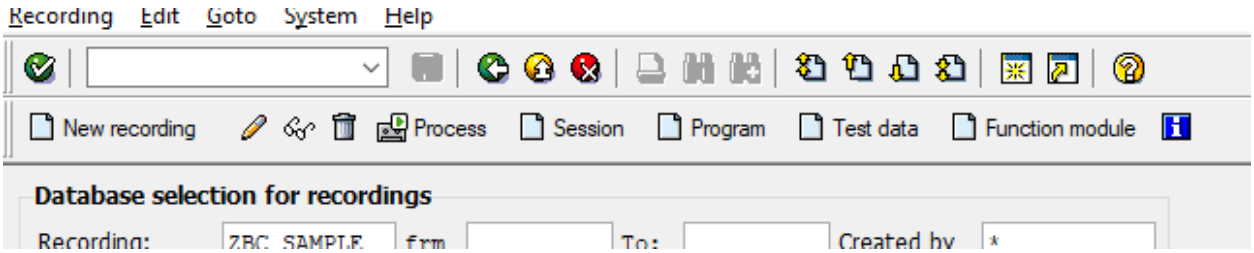
Save it on $TMP Package. click SAVE icon



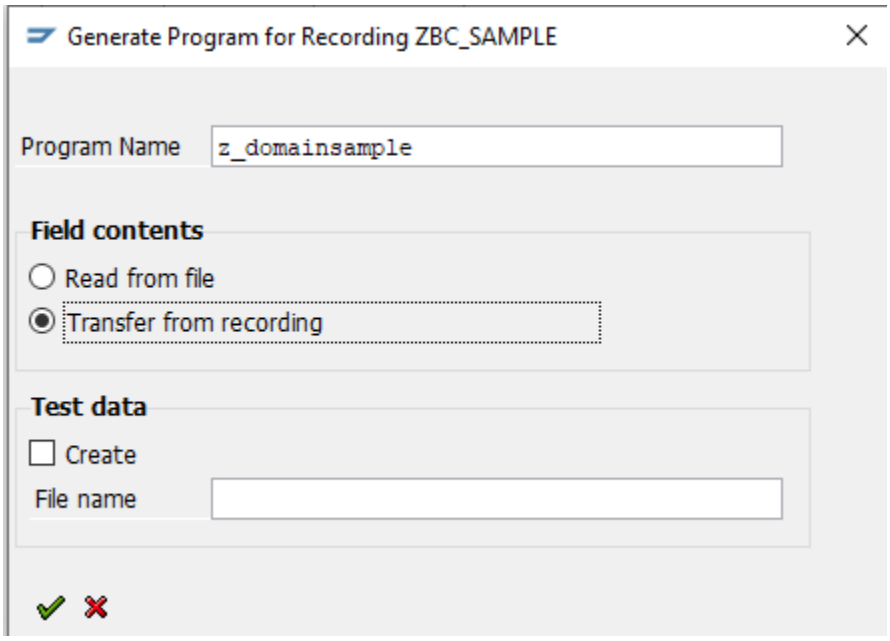
The Recording will appear at the database



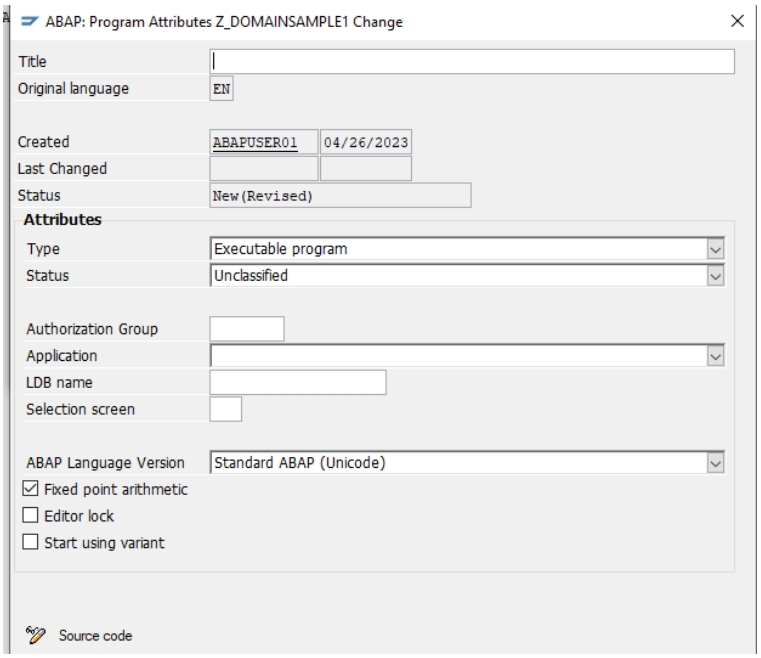
From the toolbar you will click the "Program"



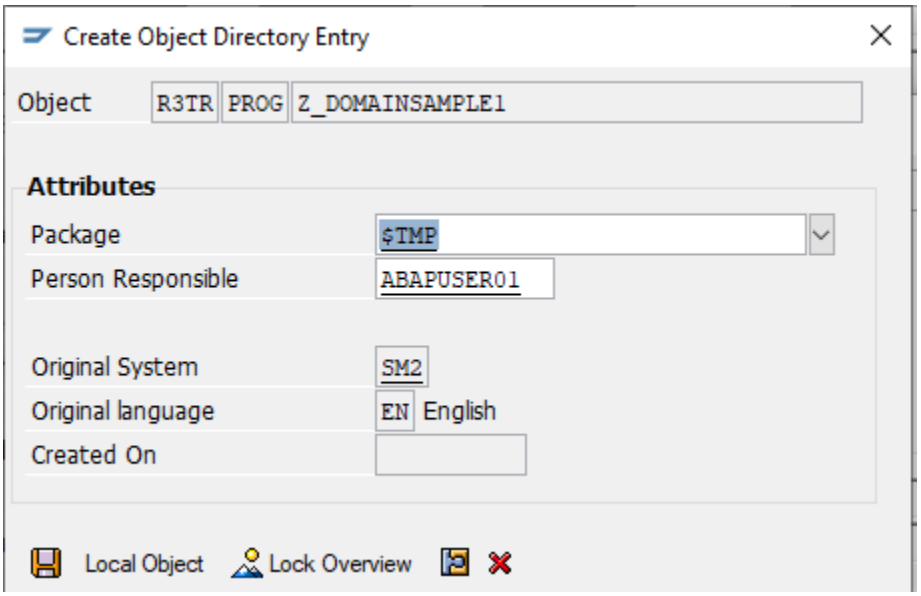
Put the program name and select the "Transfer From Recording"



Put a title on the program. Click the “Source code”



Click on save



It will redirect to the ABAP Editor with the code generated from the BDC Recording

```
report Z_DOMAINSAMPLE1
       no standard page heading line-size 255.

* Include bdcrecx1_s:
* The call transaction using is called WITH AUTHORITY-CHECK!
* If you have own auth.-checks you can use include bdcrecx1 instead.
include bdcrecx1_s.

start-of-selection.

perform open_group.

perform bdc_dynpro      using 'SAPLSD_ENTRY' '1000'.
perform bdc_field       using 'BDC_CURSOR'
                              'RSRD1-DOMA_VAL'.
perform bdc_field       using 'BDC_OKCODE'
                              '=WB_CREATE'.
perform bdc_field       using 'RSRD1-DOMA'
                              'X'.
perform bdc_field       using 'RSRD1-DOMA_VAL'
                              'ZDOMAINsample1'.
perform bdc_dynpro      using 'SAPLSD11' '1200'.
perform bdc_field       using 'BDC_OKCODE'
                              '/00'.
perform bdc_field       using 'DD01D-DDTEXT'
                              'Sample 1'.
perform bdc_field       using 'BDC_CURSOR'
                              'DD01D-OUTPUTLEN'.
perform bdc_field       using 'DD01D-DATATYPE'
                              'CHAR'.
perform bdc_field       using 'DD01D-LENG'
                              '    10'.
perform bdc_field       using 'DD01D-DECIMALS'
                              '     0'.
perform bdc_field       using 'DD01D-OUTPUTLEN'
                              '    10'.
perform bdc_dynpro      using 'SAPLSD11' '1200'.
perform bdc_field       using 'BDC_OKCODE'
                              '=WB_SAVE'.
perform bdc_field       using 'DD01D-DDTEXT'
                              'Sample 1'.
perform bdc_field       using 'BDC_CURSOR'
                              'DD01D-OUTPUTLEN'.
perform bdc_field       using 'DD01D-DATATYPE'
                              'CHAR'.
perform bdc_field       using 'DD01D-LENG'
                              '    10'.
perform bdc_field       using 'DD01D-OUTPUTLEN'
                              '    10'.
perform bdc_dynpro      using 'SAPLSTRD' '0100'.
perform bdc_field       using 'BDC_CURSOR'
                              'KO007-L DEVCLASS'.
```

Declaring Truxs for Type-pools for our Conversion of XLS to Internal Table

```
TYPE-POOLS: truxs.
```

Declare the Data for our table that will be populated by excel that we will upload

```
TYPES: BEGIN OF t_datatab,
         col1(30)  TYPE c,
         col2(30)  TYPE c,
         col3(30)  TYPE c,
         col4(30)  TYPE c,
         col5(30)  TYPE c,
         col6(480) TYPE c,
       END OF t_datatab.
```

Declare the internal table where data from excel will be saved

```
DATA: it_datatab TYPE STANDARD TABLE OF t_datatab.

DATA: it_raw TYPE truxs_t_text_data.
```

Data for the Batch Input Data of single transaction

```
DATA: bdcdata    TYPE STANDARD TABLE OF bdcdata,    "OCCURS 0 WITH HEADER LINE.
      wa_bdcdata TYPE bdcdata.
```

table for the Messages(Error) of call transaction

```
DATA:   messtab TYPE STANDARD TABLE OF bdcmsgcoll.  " OCCURS 0 WITH HEADER LINE.
```

Data for Display structure

First is the Processing mode for call transaction N for Normal

Second is the Update mode for call transaction a for automatic

And v_message that we will use later for our message table

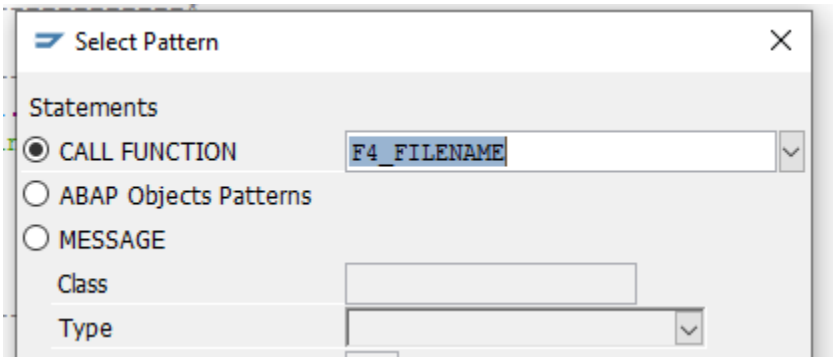
```
DATA: ctumode LIKE ctu_params-dismode VALUE 'N'.
DATA: cupdate LIKE ctu_params-updmode VALUE 'A'.
DATA: v_message(480).
```

Next is the declaration of SELECTION-SCREEN for our File Parameter.

You can change the text on selection screen by clicking F5 and going to selection text tab.

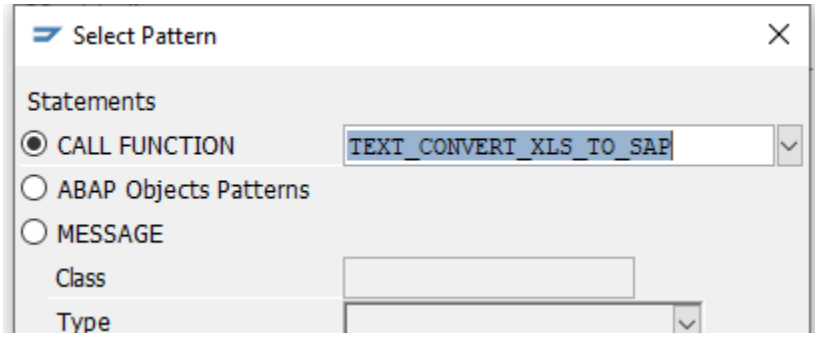
```
SELECTION-SCREEN BEGIN OF BLOCK b1 WITH FRAME TITLE TEXT-001.
  PARAMETERS: p_file TYPE  rlgrap-filename DEFAULT 'C:\Domain_Upload.xlsx'.
SELECTION-SCREEN END OF BLOCK b1.
```

AT-SELECTION SCREEN VALUE REQUEST FOR file, you will go to PATTERN and call F4_FILENAME




```
AT SELECTION-SCREEN ON VALUE-REQUEST FOR p_file.
  CALL FUNCTION 'F4_FILENAME'
    EXPORTING
      field_name = 'P_FILE'
    IMPORTING
      file_name  = p_file.
```

On our START-OF-SELECTION FUNCTION go to Pattern and Call 'TEXT_CONVERT_XLS_TO_SAP'



We are using the it_raw for the raw table data

P_file for the file parameter

It_datatab for our the destination of the converted data

```
START-OF-SELECTION.

  CALL FUNCTION 'TEXT_CONVERT_XLS_TO_SAP'
    EXPORTING
*     I_FIELD_SEPERATOR    =
      i_line_header        = 'X'
      i_tab_raw_data       = it_raw       " WORK TABLE
      i_filename           = p_file
    TABLES
      i_tab_converted_data = it_datatab[]     "ACTUAL DATA
    EXCEPTIONS
      conversion_failed    = 1
      OTHERS               = 2.

  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.

```

At the END-OF-SELECTION

We will the table using WRITE and vline for vertical lines and data from the it_datatab

```
END-OF-SELECTION.
*Write Header Column Name
  ULINE 1(255).
  WRITE:/1 sy-vline,  'Domain Name'(c01), 15 sy-vline,
         16 'Description'(c02), 50 sy-vline,
         51 'Data Type'(c03), 60 sy-vline,
         61 'No Characters'(c04), 75 sy-vline,
         76 'Decimal Places'(c05), 90 sy-vline,
         91 'Log Details'(c06), 255 sy-vline.
*SKIP.
  ULINE (255).
  LOOP AT it_datatab INTO DATA(wa_datatab).
```

Clear first the message table named messtab[]

The batch input data is stored in the bdcdata internal table, which is later used to call the transaction SE11 to create domains.

The processing mode we have declare above which is the ctumode, that has normal listing, and update that has automatic update

Messages from the message table

```
CLEAR messtab[].
CALL TRANSACTION 'SE11' WITH AUTHORITY-CHECK USING bdcdata
                        MODE   ctumode
                        UPDATE cupdate
                        MESSAGES INTO messtab.
```

We will display message from wa_messtab and save it in v_message

```
LOOP AT messtab INTO DATA(wa_messtab).
  MESSAGE ID     wa_messtab-msgid
          TYPE   wa_messtab-msgtyp
          NUMBER wa_messtab-msgnr
          INTO v_message
          WITH wa_messtab-msgv1
               wa_messtab-msgv2
               wa_messtab-msgv3
               wa_messtab-msgv4.

  IF sy-tabix = 1.
    wa_datatab-col6 = v_message.
  ELSE.
    CONCATENATE wa_datatab-col6 ',' v_message INTO wa_datatab-col6.
  ENDIF.

  CLEAR v_message.
ENDLOOP.
CLEAR bdcdata[].
```

To display the output of the program we use write and get the data from our work area data table which is wa_datatab and separated by vertical line from sy-vline.

```
*    ULINE 2(255).
    WRITE:/1 sy-vline,  wa_datatab-col1, 15 sy-vline,
           16 wa_datatab-col2, 50 sy-vline,
           51 wa_datatab-col3, 60 sy-vline,
           61 wa_datatab-col4, 75 sy-vline,
           76 wa_datatab-col5, 90 sy-vline,
           91 wa_datatab-col6, 255 sy-vline.
    ULINE AT /1(255).
  ENDLOOP.
```

It will display like

|Column 1|Column 2|…

We using this subroutine for starting new screen

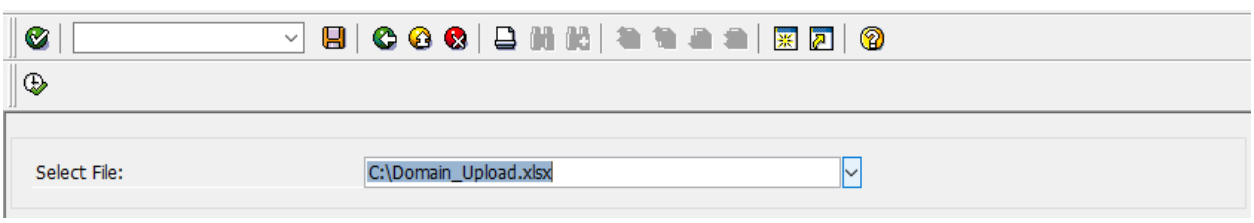
```
FORM bdc_dynpro USING program dynpro.
  CLEAR wa_bdcdata.
  wa_bdcdata-program  = program.
  wa_bdcdata-dynpro   = dynpro.
  wa_bdcdata-dynbegin = 'X'.
  APPEND wa_bdcdata TO bdcdata.
ENDFORM.
```

This subroutine is used for appending a new line of data to an internal table called "bdcdata"

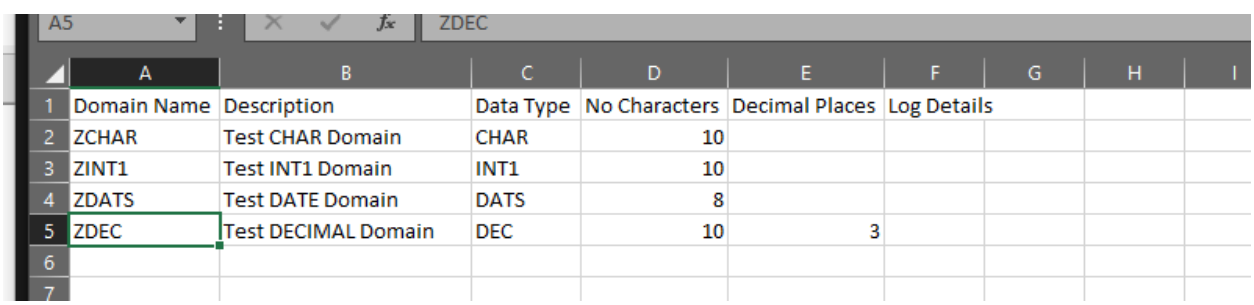
```
FORM bdc_field USING fnam fval.
*  IF fval <> nodata.
  CLEAR wa_bdcdata.
  wa_bdcdata-fnam = fnam.
  wa_bdcdata-fval = fval.
  APPEND wa_bdcdata TO bdcdata.
*  ENDIF.
ENDFORM.
```

Output:

File Selection:

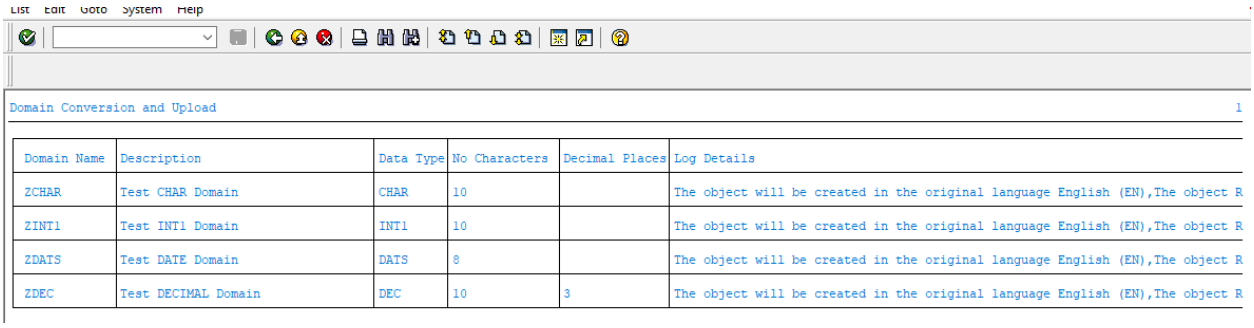


XLS File:



	A	B	C	D	E	F
1	Domain Name	Description	Data Type	No Characters	Decimal Places	Log Details
2	ZCHAR	Test CHAR Domain	CHAR	10		
3	ZINT1	Test INT1 Domain	INT1	10		
4	ZDATS	Test DATE Domain	DATS	8		
5	ZDEC	Test DECIMAL Domain	DEC	10	3	
6						
7						

Result:



List Edit Goto System Help

Domain Conversion and Upload 1

Domain Name	Description	Data Type	No Characters	Decimal Places	Log Details
ZCHAR	Test CHAR Domain	CHAR	10		The object will be created in the original language English (EN),The object R
ZINT1	Test INT1 Domain	INT1	10		The object will be created in the original language English (EN),The object R
ZDATS	Test DATE Domain	DATS	8		The object will be created in the original language English (EN),The object R
ZDEC	Test DECIMAL Domain	DEC	10	3	The object will be created in the original language English (EN),The object R