Introduction to Data Migration

1. Introduction

- Data Migration is also called as data transfer. Data Migration means migrating the data from legacy system to SAP System.

  ### What is Legacy System ?

  - The system other than the SAP system is called as legacy system.
  - It is also called as Non-SAP system.
  - It contains the data to be transferred.

E.g.

- Suppose, One organization is working on some other technologies like Java, C++ etc. and now, they want to transfer their business to SAP i.e. they want to implement SAP for their business, So In that situation our old system to legacy system and we will migrate our data to SAP system.

---

2. Steps for Data Migration

1. Extracting the data :- Extracting the data from the legacy system into a file.
2. Converting the File :- Converting the data to appropriate format that is supported by SAP. This is called as Conversion.
3. Importing the data :- Importing the data to SAP System
4. Verifying the data :- Checking the accuracy of data in SAP system.

---

3. Data Migration Techniques

1. For Technical Consultant

1. BDC(Batch data Communication)
2. BAPI(Business Application Programming Interface)

2. For Functional Consultant

1. LSMW(Legacy System Migration Workbench)

BDC(Batch Data Communication) in ABAP

1. Introduction

- BDC stands for Batch Data Communication
- The purpose of BDC is to transfer data from Non-SAP(Legacy) system to SAP system.
- BDC works on the principle of Screen recording, i.e. We can record the steps of Implementation and then we can reuse it in the program for large scale migration.
- The transaction code for Recording is SHDB.

Precaution for Recording

1. Never Use F4 and F1 Help.
2. Do not pass wrong values.

BDC Methods

1. Call Transaction Method
2. Session Method

3. Direct Input Method

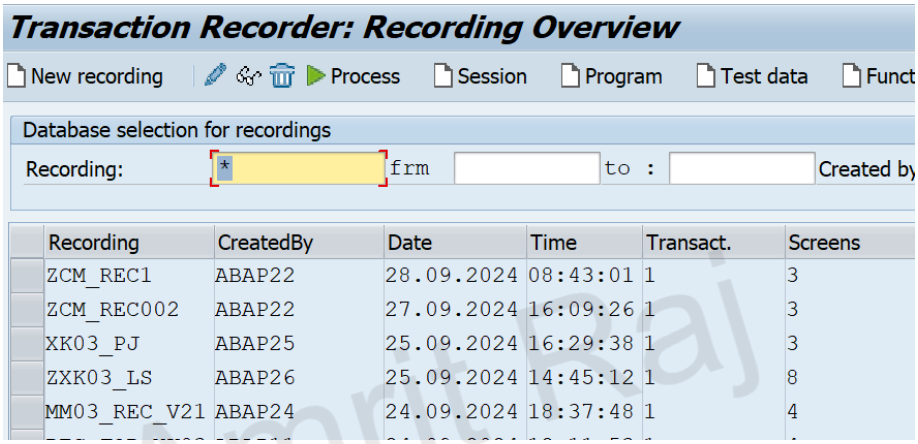
Note :-

- Call transaction method and session method are called as Batch Input Methods.

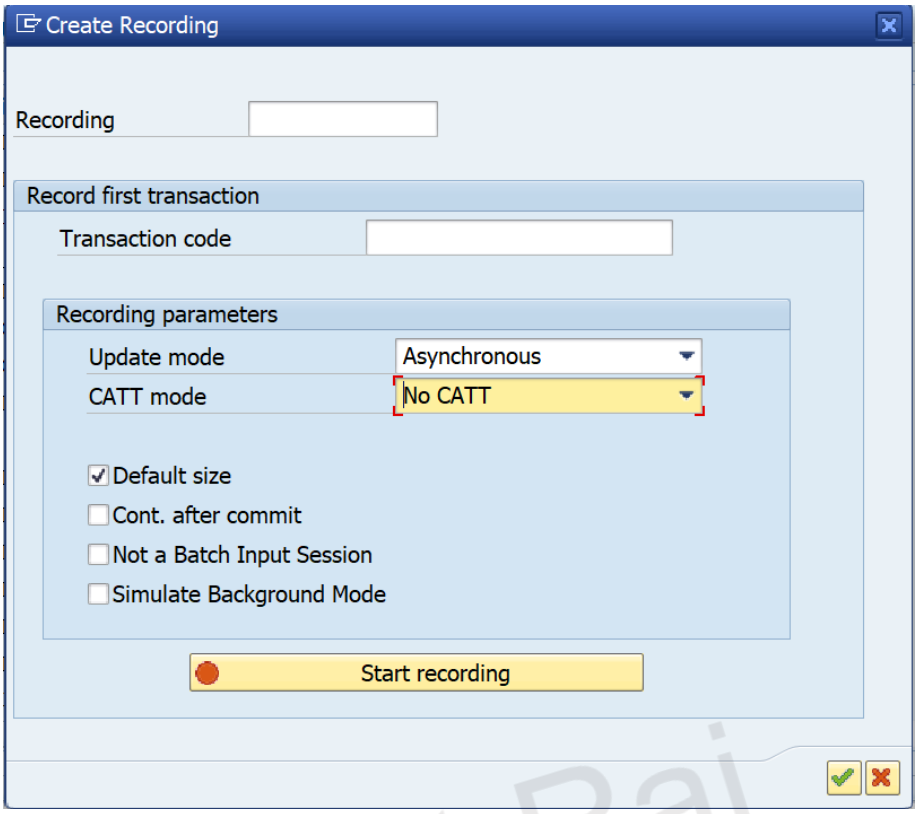
---

2. Screen Recording for Creating Material

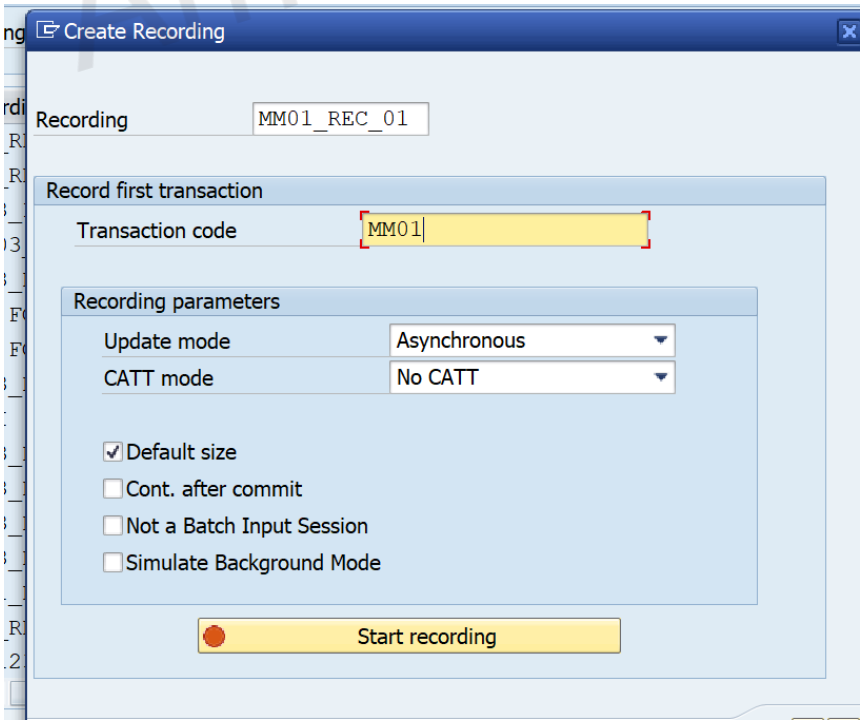
- Step 1 :- Go to transaction code SHDB.



- Step 2 :- Click on new Recording.

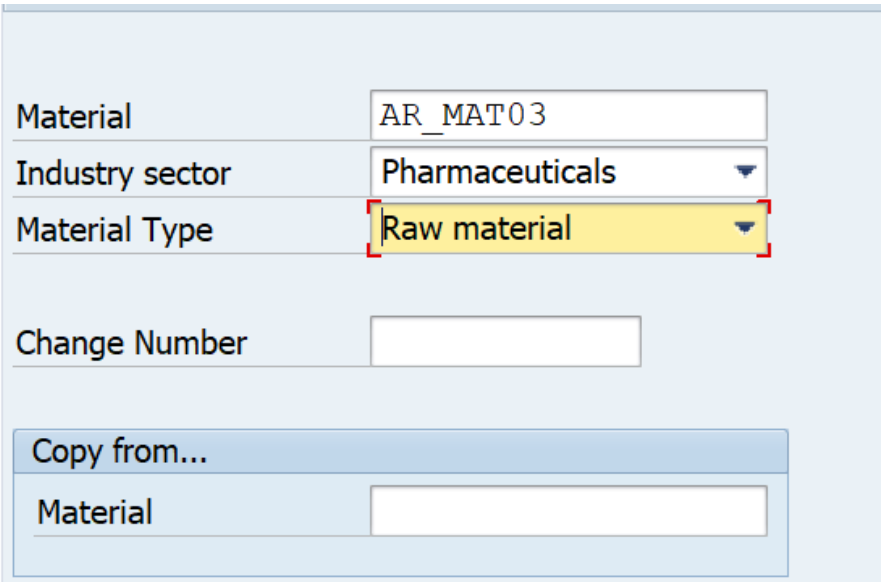


- Provide the name of your recording and give the transaction code.

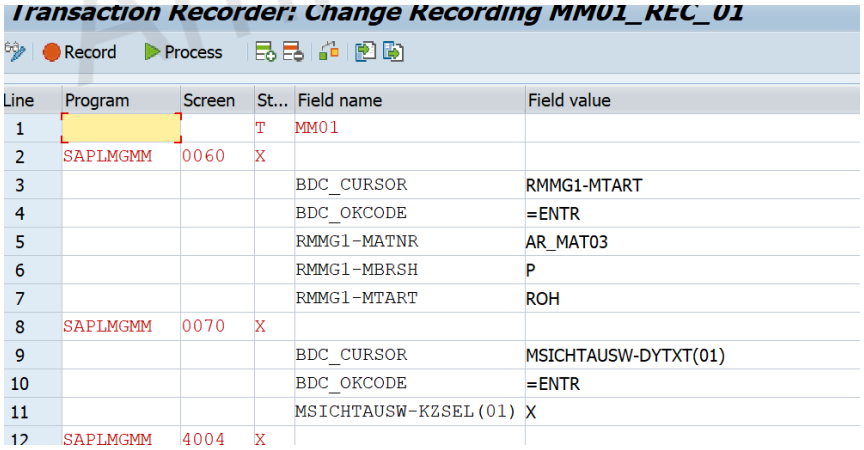


Mob No :- +916261538504

- Step 3 :- Click on start recording and you will navigate to mm01.

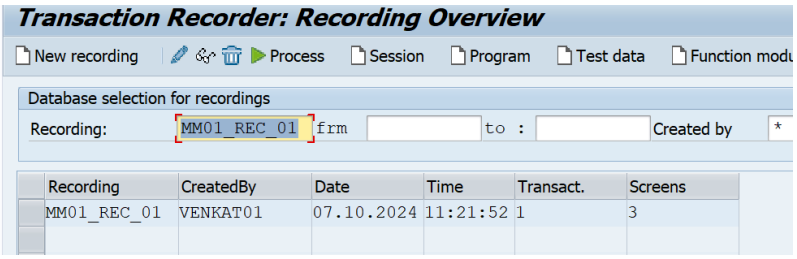


  - Press Enter and select Basic data 1 and click on Okay button.
  - Provide the short description and basic unit of measurement as EU and click on save button.

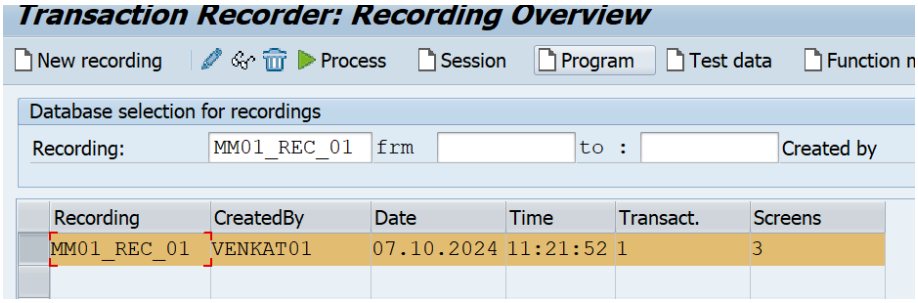


Line	Program	Screen	St...	Field name	Field value
1			T	MM01	
2	SAPLMGMM	0060	X		
3				BDC_CURSOR	RMMG1-MTART
4				BDC_OKCODE	=ENTR
5				RMMG1-MATNR	AR_MAT03
6				RMMG1-MBRSH	P
7				RMMG1-MTART	ROH
8	SAPLMGMM	0070	X		
9				BDC_CURSOR	MSICHTAUSW-DYTXT(01)
10				BDC_OKCODE	=ENTR
11				MSICHTAUSW-KZSEL(01)	X
12	SAPLMGMM	4004	X		

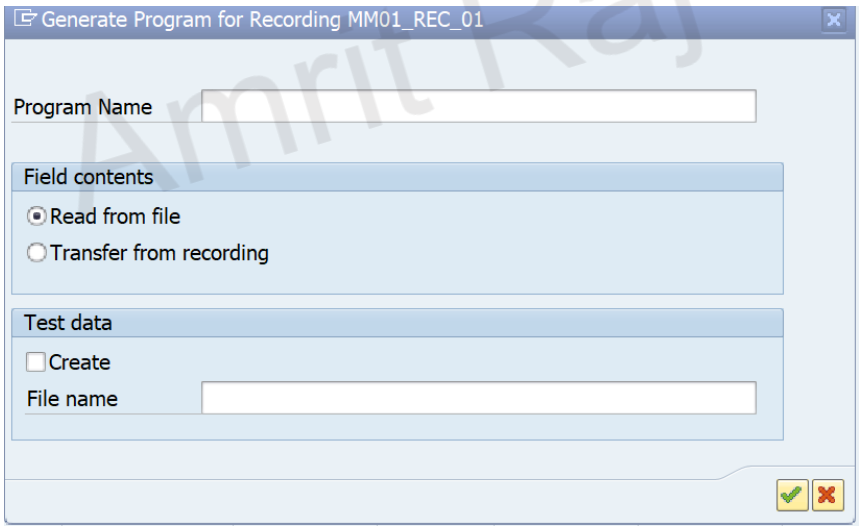
    - Click on back button and we will see our recording is ready.



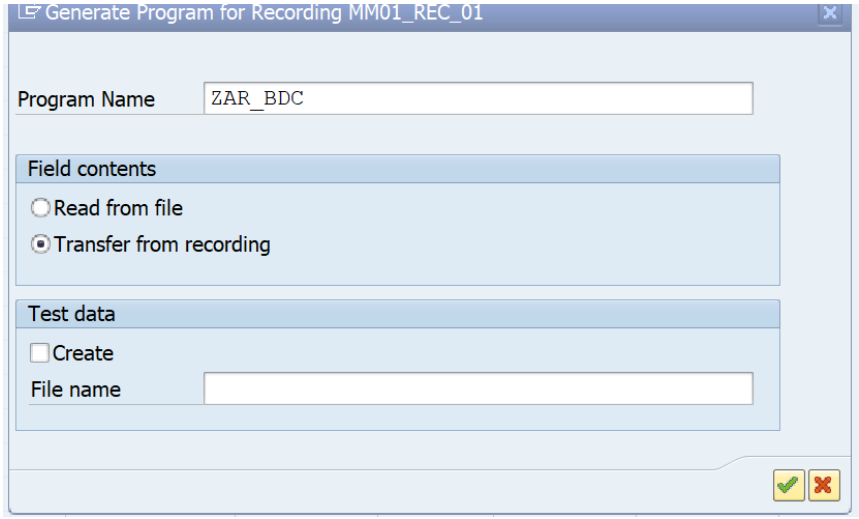
- Step 4 :- Select the recording.



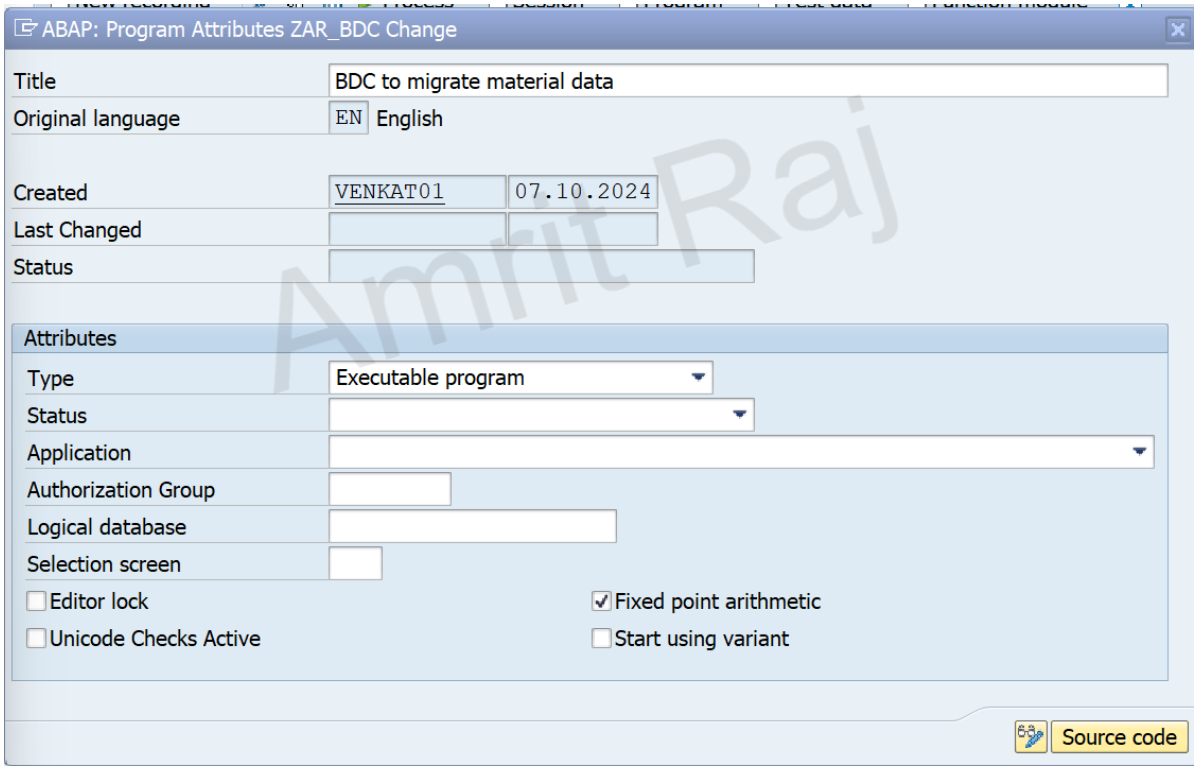
  - Click on Program and a pop up screen will appear.



  - Give a program name and Select Transfer from recording radio button.



- Step 5 :- Click on Okay button and provide a title for our program.

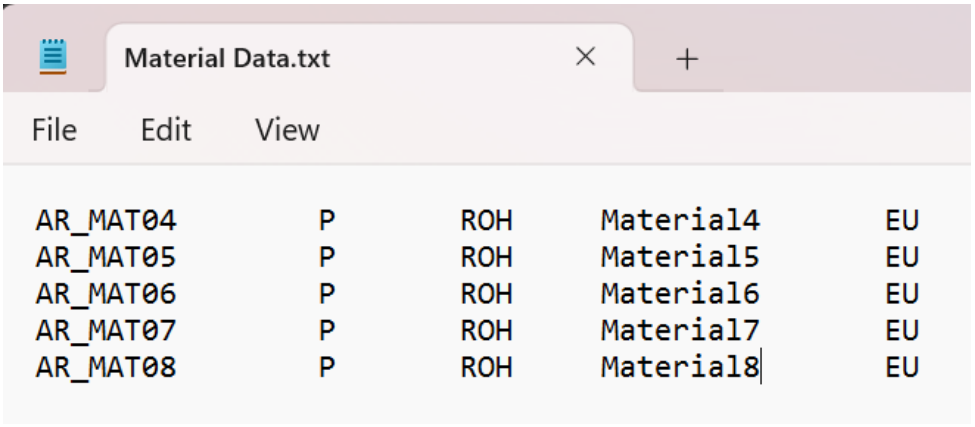


  - Click on source code and assign the package and transport request.

```
port            ZAR_BDC                                  Active
 1  report ZAR_BDC
 2         no standard page heading line-size 255.
 3
 4  include bdcrecx1.
 5
 6  start-of-selection.
 7
 8  perform open_group.
 9
10  perform bdc_dynpro      using 'SAPLMGMM' '0060'.
11  perform bdc_field       using 'BDC_CURSOR'
12                                'RMMG1-MTART'.
13  perform bdc_field       using 'BDC_OKCODE'
14                                '=ENTR'.
15  perform bdc_field       using 'RMMG1-MATNR'
16                                'AR_MAT03'.
17  perform bdc_field       using 'RMMG1-MBRSH'
18                                'P'.
19  perform bdc_field       using 'RMMG1-MTART'
20                                'ROH'.
21  perform bdc_dynpro      using 'SAPLMGMM' '0070'.
22  perform bdc_field       using 'BDC_CURSOR'
23                                'MSICHTAUSW-DYTXT(01)'.
24  perform bdc_field       using 'BDC_OKCODE'
25                                '=ENTR'.
26  perform bdc_field       using 'MSICHTAUSW-KZSEL(01)'
27                                'X'.
28  perform bdc_dynpro      using 'SAPLMGMM' '4004'.
29  perform bdc_field       using 'BDC_OKCODE'
30                                '=BU'.
31  perform bdc_field       using 'MAKT-MAKTX'
```

- Our program is ready.

3. Data that we will Migrate



- Above is the 5 records that we will migrate, the various columns that we can see here are Material Number, Industry Sector, Industry Type, Material Description, Basic unit of measure etc.

4. Implementing the BDC Program

- Step 1 :- First we need to bring the notepad file into our local system so for that purpose we will create a parameter of type local file and before that we will define a type structure and internal table and work area for the file as well.

```
5   TYPES : BEGIN OF TY_MARA,
6     MATNR type MATNR,
7     MBRSH type MBRSH,
8     MTART type MTART,
9     MAKTX type MAKTX,
10    MEINS type MEINS,
11    END OF TY_MARA.
12
13  DATA : LT_MARA TYPE TABLE OF TY_MARA,
14         LS_MARA TYPE TY_MARA.
15
16  PARAMETERS : P_FILE TYPE LOCALFILE.
17
```

- Step 2 :- We will use At Selection Screen on Value Request event to apply a F4 Help and then we will use F4_FILENAME function module to pick local file from our PC.

```
18  AT SELECTION-SCREEN ON VALUE-REQUEST FOR P_FILE.
19    CALL FUNCTION 'F4_FILENAME'
20      EXPORTING
21        PROGRAM_NAME  = SYST-CPROG
22        DYNPRO_NUMBER = SYST-DYNNR
23        FIELD_NAME    = ' '
24      IMPORTING
25        FILE_NAME     = P_FILE.
26
27
```

- Step 3 :- Now to upload this file to SAP System we will use GUI_UPLOAD function module to upload file into the SAP System which will return the data into our desired internal table.

Mob No :- +916261538504

```
START-OF-SELECTION.
LO_FILE = P_FILE.
  CALL FUNCTION 'GUI_UPLOAD'
    EXPORTING
      FILENAME                = LO_FILE
*     FILETYPE                = 'ASC'
      HAS_FIELD_SEPARATOR     = 'X'
    TABLES
      DATA_TAB                = LT_MARA
* CHANGING
*     ISSCANPERFORMED         = ' '
    EXCEPTIONS
      FILE_OPEN_ERROR         = 1
      FILE_READ_ERROR         = 2
      NO_BATCH                = 3
      GUI_REFUSE_FILETRANSFER = 4
      INVALID_TYPE            = 5
      NO_AUTHORITY            = 6
      UNKNOWN_ERROR           = 7
      BAD_DATA_FORMAT         = 8
      HEADER_NOT_ALLOWED      = 9
      SEPARATOR_NOT_ALLOWED   = 10
      HEADER_TOO_LONG         = 11
      UNKNOWN_DP_ERROR        = 12
      ACCESS_DENIED           = 13
      DP_OUT_OF_MEMORY        = 14
      DISK_FULL               = 15
      DP_TIMEOUT              = 16
      OTHERS                  = 17.
```

- Step 4 :- Then we will un comment the rest of the logic except the start of selection part.

```
include bdcrecx1.
*
*start-of-selection.
*
perform open_group.

perform bdc_dynpro      using 'SAPLMGMM' '0060'.
perform bdc_field       using 'BDC_CURSOR'
                              'RMMG1-MTART'.
perform bdc_field       using 'BDC_OKCODE'
                              '=ENTR'.
perform bdc_field       using 'RMMG1-MATNR'
                              'AR_MAT03'.
perform bdc_field       using 'RMMG1-MBRSH'
                              'P'.
perform bdc_field       using 'RMMG1-MTART'
                              'ROH'.
perform bdc_dynpro      using 'SAPLMGMM' '0070'.
perform bdc_field       using 'BDC_CURSOR'
                              'MSICHTAUSW-DYTXT(01)'.
perform bdc_field       using 'BDC_OKCODE'
                              '=ENTR'.
perform bdc_field       using 'MSICHTAUSW-KZSEL(01)'
                              'X'.
perform bdc_dynpro      using 'SAPLMGMM' '4004'.
perform bdc_field       using 'BDC_OKCODE'
```

- Step 5 :- Now we will use the a loop statement to loop for all our records.

```
66
67
68  LOOP AT LT_MARA INTO LS_MARA.
69    PERFORM OPEN_GROUP.
70
71    PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '0060'.
72    PERFORM BDC_FIELD       USING 'BDC_CURSOR'
73                                  'RMMG1-MTART'.
74    PERFORM BDC_FIELD       USING 'BDC_OKCODE'
75                                  '=ENTR'.
76    PERFORM BDC_FIELD       USING 'RMMG1-MATNR'
77                                  'AR_MAT03'.
78    PERFORM BDC_FIELD       USING 'RMMG1-MBRSH'
```

- Step 6 :- We will comment the perform OPEN_GROUP, BDC transaction and perform CLOSE_GROUP.

```
67
68  LOOP AT LT_MARA INTO LS_MARA.
69  *   PERFORM OPEN_GROUP.
70
71    PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '0060'.
72    PERFORM BDC_FIELD       USING 'BDC_CURSOR'
73                                  'RMMG1-MTART'.
74    PERFORM BDC_FIELD       USING 'BDC_OKCODE'
75                                  '=ENTR'.
76    PERFORM BDC_FIELD       USING 'RMMG1-MATNR'
77                                  'AR_MAT03'.
78    PERFORM BDC_FIELD       USING 'RMMG1-MBRSH'
79                                  'P'.
80    PERFORM BDC_FIELD       USING 'RMMG1-MTART'
81                                  'ROH'.
82    PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '0070'.
83    PERFORM BDC_FIELD       USING 'BDC_CURSOR'
84                                  'MSICHTAUSW-DYTXT(01)'.
85    PERFORM BDC_FIELD       USING 'BDC_OKCODE'
86                                  '=ENTR'.
87    PERFORM BDC_FIELD       USING 'MSICHTAUSW-KZSEL(01)'
88                                  'X'.
89    PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '4004'.
90    PERFORM BDC_FIELD       USING 'BDC_OKCODE'
91                                  '=BU'.
92    PERFORM BDC_FIELD       USING 'MAKT-MAKTX'
93                                  'Material Description'.
94    PERFORM BDC_FIELD       USING 'BDC_CURSOR'
95                                  'MARA-MEINS'.
96    PERFORM BDC_FIELD       USING 'MARA-MEINS'
97                                  'EU'.
98  *   PERFORM BDC_TRANSACTION USING 'MM01'.
99
100 *   PERFORM CLOSE_GROUP.
101
102   ENDLOOP.
```


- Step 7 :- Now, Inside our loop we can see we have two perform BDC_DYNPRO and BDC_FIELD, so for that we are required to created Form (Subroutines) as well, so for that purpose, we will go inside this below include.

```
60
61
62      INCLUDE BDCRECX1.
63  *
64  *start-of-selection.
65  *
```

- Double click on the include and click on display button, and at the end you will be able to see the below subroutines, so simply copy them and paste in your programs.

```
237 *---------------------------------------
238 FORM BDC_DYNPRO USING PROGRAM DYNPRO.
239   CLEAR BDCDATA.
240   BDCDATA-PROGRAM  = PROGRAM.
241   BDCDATA-DYNPRO   = DYNPRO.
242   BDCDATA-DYNBEGIN = 'X'.
243   APPEND BDCDATA.
244 ENDFORM.
245
246 *---------------------------------------
247 *        Insert field
248 *---------------------------------------
249 FORM BDC_FIELD USING FNAM FVAL.
250   IF FVAL <> NODATA.
251     CLEAR BDCDATA.
252     BDCDATA-FNAM = FNAM.
253     BDCDATA-FVAL = FVAL.
254     APPEND BDCDATA.
255   ENDIF.
256 ENDFORM.
257
```

- Step 8 :- Copy the Subroutines and paste in our programs.

```
103
104   FORM BDC_DYNPRO USING PROGRAM DYNPRO.
105   CLEAR BDCDATA.
106   BDCDATA-PROGRAM  = PROGRAM.
107   BDCDATA-DYNPRO   = DYNPRO.
108   BDCDATA-DYNBEGIN = 'X'.
109   APPEND BDCDATA.
110 ENDFORM.
111
112 *---------------------------------------
113 *        Insert field
114 *---------------------------------------
115 FORM BDC_FIELD USING FNAM FVAL.
116   IF FVAL <> NODATA.
117     CLEAR BDCDATA.
118     BDCDATA-FNAM = FNAM.
119     BDCDATA-FVAL = FVAL.
120     APPEND BDCDATA.
121   ENDIF.
122 ENDFORM.
123
```

- Step 9 :- Now, in these subroutines we can see that SAP is simply appending data in BDCDATA, so we will simply check the type and we will create a internal table and work area accordingly.

```
13   DATA : LT_MARA TYPE TABLE OF TY_MARA,
14          LS_MARA TYPE TY_MARA.
15
16   DATA : LT_BDCDATA type table of BDCDATA,
17          LS_BDCDATA type BDCDATA.
18   DATA : LO_FILE TYPE STRING.
19   PARAMETERS : P_FILE TYPE LOCALFILE.
20
```

- Step 10 :- We will comment that include.

```
61
62
63
64
65   *   INCLUDE BDCRECX1.
66   *
67   *start-of-selection.
68   *
69
```


- Step 11 :- Now we will go to the subroutines and check the internal table and work area accordingly, and also remove the if statement from the BDC_FIELD subroutine.

```
107    FORM BDC_DYNPRO USING PROGRAM DYNPRO.
108    CLEAR LS_BDCDATA.
109    LS_BDCDATA-PROGRAM  = PROGRAM.
110    LS_BDCDATA-DYNPRO   = DYNPRO.
111    LS_BDCDATA-DYNBEGIN = 'X'.
112    APPEND LS_BDCDATA to LT_BDCDATA.
113  ENDFORM.
114
115  *---------------------------------------
116  *        Insert field
117  *---------------------------------------
118  FORM BDC_FIELD USING FNAM FVAL.
119      CLEAR LS_BDCDATA.
120      LS_BDCDATA-FNAM = FNAM.
121      LS_BDCDATA-FVAL = FVAL.
122      APPEND LS_BDCDATA to LT_BDCDATA.
123  ENDFORM.
```

- Step 12 :- Then we will pass the data of our work area into the into the perform.

```
70
71    LOOP AT LT_MARA INTO LS_MARA.
72  *     PERFORM OPEN_GROUP.
73        PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '0060'.
74        PERFORM BDC_FIELD       USING 'BDC_CURSOR'
75                                      'RMMG1-MTART'.
76        PERFORM BDC_FIELD       USING 'BDC_OKCODE'
77                                      '=ENTR'.
78        PERFORM BDC_FIELD       USING 'RMMG1-MATNR'
79                                      LS_MARA-MATNR.
80        PERFORM BDC_FIELD       USING 'RMMG1-MBRSH'
81                                      LS_MARA-MBRSH.
82        PERFORM BDC_FIELD       USING 'RMMG1-MTART'
83                                      LS_MARA-MTART.
84        PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '0070'.
85        PERFORM BDC_FIELD       USING 'BDC_CURSOR'
86                                      'MSICHTAUSW-DYTXT(01)'.
87        PERFORM BDC_FIELD       USING 'BDC_OKCODE'
88                                      '=ENTR'.
89        PERFORM BDC_FIELD       USING 'MSICHTAUSW-KZSEL(01)'
90                                      'X'.
91        PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '4004'.
92        PERFORM BDC_FIELD       USING 'BDC_OKCODE'
93                                      '=BU'.
94        PERFORM BDC_FIELD       USING 'MAKT-MAKTX'
95                                      LS_MARA-MAKTX.
96        PERFORM BDC_FIELD       USING 'BDC_CURSOR'
97                                      'MARA-MEINS'.
98        PERFORM BDC_FIELD       USING 'MARA-MEINS'
99                                      LS_MARA-MEINS.
100 *     PERFORM BDC_TRANSACTION USING 'MM01'.
101 *     PERFORM CLOSE_GROUP.
102   ENDLOOP.
103
```


5. Call Transaction Method

Syntax :-

- CALL TRANSACTION ‘TRANSACTION CODE’ USING (BDC Internal Table) MODE ‘A or N or E’ UPDATE ‘A or S or L’ Messages INTO (Internal Table).

  Example :- CALL TRANSACTION ‘MM01’ USING LT_BTDCDATA MODE ‘A’ UPDATE ‘S’ Messages INTO LT_MESSTAB.

Processing Nodes

1. A → All Screen (It will show screen by screen processing)
2. N → No Screen (It will not show any screen, directly we will get a output).

3. E → Error (If there is a error in BDC - It will show the screen where the error is, If there is no error - It will not show any screen).

Update Nodes

1. A → Asynchronous(COMMIT WORK)
   - The called transaction does not wait for any updates to be completed.
   - Results in faster execution of the data transfer program.
2. S → Synchronous(Sync between the sender and receiver, COMMIT WORK and WAIT)
   - This called transaction waits for any updates than it produces to be completed.
   - Execution is slower than with asynchronous updating because called transactions wait for updating to be completed.
3. L → Local Update
   - If the data is updated locally, the update of the database will not be processed in a separate process, the update functions are run in the same dialog process.

Implementation

- Step 1 :- Just before the end of loop we will call the Call transaction method using the internal table of BDCDATA.

```
    PERFORM BDC_FIELD       USING 'MARA-MEINS'
                                  LS_MARA-MEINS.
*     PERFORM BDC_TRANSACTION USING 'MM01'.
*     PERFORM CLOSE_GROUP.

    CALL TRANSACTION 'MM01' USING LT_BDCDATA.
  ENDLOOP.

  FORM BDC_DYNPRO USING PROGRAM DYNPRO.
```

- Step 2 :- We will use the processing node, update nodes and message internal table to process all the data.
  - For the message tab we will go again to the include to fetch its type.

```
18
19  DATA : LT_MESSAGE type table of BDCMSGCOLL,
20         LS_MESSAGE type BDCMSGCOLL.
21
```

```
109    CALL TRANSACTION 'MM01' USING LT_BDCDATA MODE 'A' UPDATE 'S' MESSAGES INTO LT_MESSAGE.
110    REFRESH : LT_BDCDATA.
111  ENDLOOP.
112
113
```

- After Call transaction don’t forget to refresh the internal table of BDCDATA.

- Step 3 :- Now to see the contents of messages we will call the MESSAGE_TEXT_BUILD function module to display the messages.

```
112
113  LOOP at LT_MESSAGE into LS_MESSAGE.
114    CALL FUNCTION 'MESSAGE_TEXT_BUILD'
115      EXPORTING
116        MSGID                      = LS_MESSAGE-MSGID
117        MSGNR                      = LS_MESSAGE-MSGNR
118       MSGV1                      = LS_MESSAGE-MSGV1
119       MSGV2                      = LS_MESSAGE-MSGV2
120       MSGV3                      = LS_MESSAGE-MSGV3
121       MSGV4                      = LS_MESSAGE-MSGV4
122      IMPORTING
123        MESSAGE_TEXT_OUTPUT        = LV_MESSAGE
124               .
125
126    WRITE :/ LV_MESSAGE.
127
128  ENDLOOP.
129
```


Code

```
REPORT ZAR_BDC
       NO STANDARD PAGE HEADING LINE-SIZE 255.


TYPES : BEGIN OF TY_MARA,
  MATNR TYPE MATNR,
  MBRSH TYPE MBRSH,
  MTART TYPE MTART,
  MAKTX TYPE MAKTX,
  MEINS TYPE MEINS,
```

```
  END OF TY_MARA.

DATA : LT_MARA TYPE TABLE OF TY_MARA,
       LS_MARA TYPE TY_MARA.

DATA : LT_BDCDATA type table of BDCDATA,
       LS_BDCDATA type BDCDATA.

DATA : LT_MESSAGE type table of BDCMSGCOLL,
       LS_MESSAGE type BDCMSGCOLL.

DATA : LV_MESSAGE TYPE STRING.

DATA : LO_FILE TYPE STRING.
PARAMETERS : P_FILE TYPE LOCALFILE.

AT SELECTION-SCREEN ON VALUE-REQUEST FOR P_FILE.
  CALL FUNCTION 'F4_FILENAME'
    EXPORTING
      PROGRAM_NAME  = SYST-CPROG
      DYNPRO_NUMBER = SYST-DYNNR
      FIELD_NAME    = ' '
    IMPORTING
      FILE_NAME     = P_FILE.



START-OF-SELECTION.
  LO_FILE = P_FILE.
  CALL FUNCTION 'GUI_UPLOAD'
    EXPORTING
      FILENAME                = LO_FILE
*     FILETYPE                = 'ASC'
      HAS_FIELD_SEPARATOR     = 'X'
    TABLES
      DATA_TAB                = LT_MARA
```

```
* CHANGING
*     ISSCANPERFORMED         = ' '
    EXCEPTIONS
      FILE_OPEN_ERROR         = 1
      FILE_READ_ERROR         = 2
      NO_BATCH                = 3
      GUI_REFUSE_FILETRANSFER = 4
      INVALID_TYPE            = 5
      NO_AUTHORITY            = 6
      UNKNOWN_ERROR           = 7
      BAD_DATA_FORMAT         = 8
      HEADER_NOT_ALLOWED      = 9
      SEPARATOR_NOT_ALLOWED   = 10
      HEADER_TOO_LONG         = 11
      UNKNOWN_DP_ERROR        = 12
      ACCESS_DENIED           = 13
      DP_OUT_OF_MEMORY        = 14
      DISK_FULL               = 15
      DP_TIMEOUT              = 16
      OTHERS                  = 17.



*  INCLUDE BDCRECX1.
*
*start-of-selection.
*


  LOOP AT LT_MARA INTO LS_MARA.
*    PERFORM OPEN_GROUP.
    PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '0060'.
    PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                                  'RMMG1-MTART'.
    PERFORM BDC_FIELD       USING 'BDC_OKCODE'
```

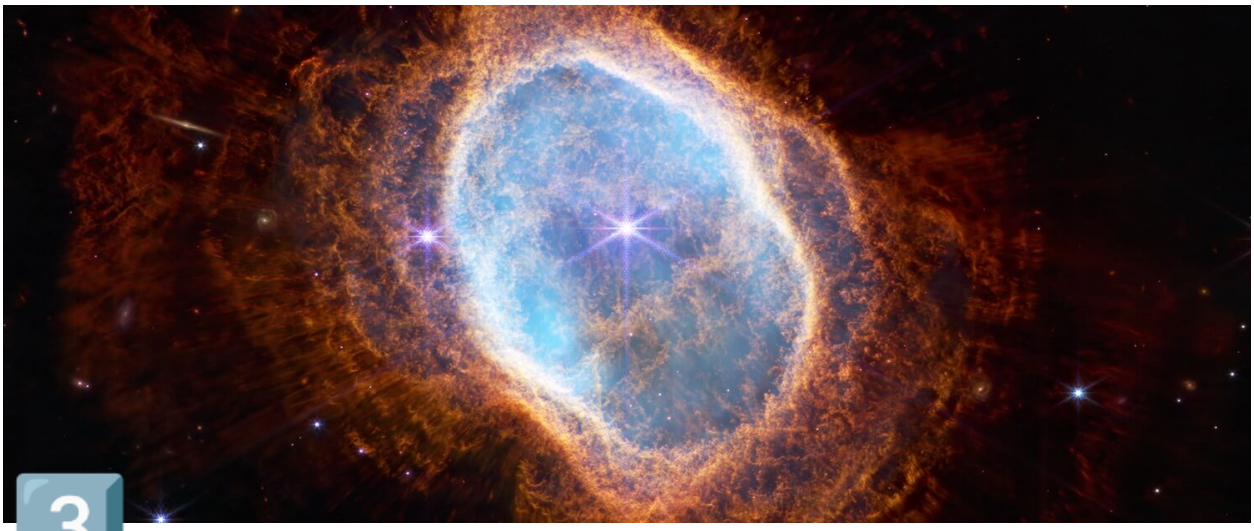
```
                                    '=ENTR'.
    PERFORM BDC_FIELD       USING 'RMMG1-MATNR'
                                    LS_MARA-MATNR.
    PERFORM BDC_FIELD       USING 'RMMG1-MBRSH'
                                    LS_MARA-MBRSH.
    PERFORM BDC_FIELD       USING 'RMMG1-MTART'
                                    LS_MARA-MTART.
    PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '0070'.
    PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                                    'MSICHTAUSW-DYTXT(01)'.
    PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                                    '=ENTR'.
    PERFORM BDC_FIELD       USING 'MSICHTAUSW-KZSEL(01)'
                                    'X'.
    PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '4004'.
    PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                                    '=BU'.
    PERFORM BDC_FIELD       USING 'MAKT-MAKTX'
                                    LS_MARA-MAKTX.
    PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                                    'MARA-MEINS'.
    PERFORM BDC_FIELD       USING 'MARA-MEINS'
                                    LS_MARA-MEINS.
*    PERFORM BDC_TRANSACTION USING 'MM01'.
*    PERFORM CLOSE_GROUP.

    CALL TRANSACTION 'MM01' USING LT_BDCDATA MODE 'A' UPDATE 'S
    REFRESH : LT_BDCDATA.
  ENDLOOP.


  LOOP at LT_MESSAGE into LS_MESSAGE.
    CALL FUNCTION 'MESSAGE_TEXT_BUILD'
      EXPORTING
        MSGID                     = LS_MESSAGE-MSGID
        MSGNR                     = LS_MESSAGE-MSGNR
```

```
        MSGV1                       = LS_MESSAGE-MSGV1
        MSGV2                       = LS_MESSAGE-MSGV2
        MSGV3                       = LS_MESSAGE-MSGV3
        MSGV4                       = LS_MESSAGE-MSGV4
      IMPORTING
        MESSAGE_TEXT_OUTPUT         = LV_MESSAGE
               .

    WRITE :/ LV_MESSAGE.

  ENDLOOP.

  FORM BDC_DYNPRO USING PROGRAM DYNPRO.
  CLEAR LS_BDCDATA.
  LS_BDCDATA-PROGRAM  = PROGRAM.
  LS_BDCDATA-DYNPRO   = DYNPRO.
  LS_BDCDATA-DYNBEGIN = 'X'.
  APPEND LS_BDCDATA to LT_BDCDATA.
ENDFORM.

*---------------------------------------------------------------------
*        Insert field
*---------------------------------------------------------------------
FORM BDC_FIELD USING FNAM FVAL.
    CLEAR LS_BDCDATA.
    LS_BDCDATA-FNAM = FNAM.
    LS_BDCDATA-FVAL = FVAL.
    APPEND LS_BDCDATA to LT_BDCDATA.
ENDFORM.
```


Output

BDC to migrate material data

```
Material AR_MAT09 created
Material AR_MAT10 created
Material AR_MAT11 created
Material AR_MAT12 created
Material AR_MAT13 created
```



Session Method in BDC

- In session method, A session will be created for your BDC program and you will have to process that session to complete the insertion part.
- In Session method we will use the following function modules :-
  1. BDC_OPEN_GROUP to open the group
  2. BDC_INSERT
  3. BDC_CLOSE_GROUP to close the group

- Step 1 :- Just before the start of the Loop we will use BDC_OPEN_GROUP function module to open a group.
  - In the group parameter we will pass a group name and a corresponding group will be created in SM35.

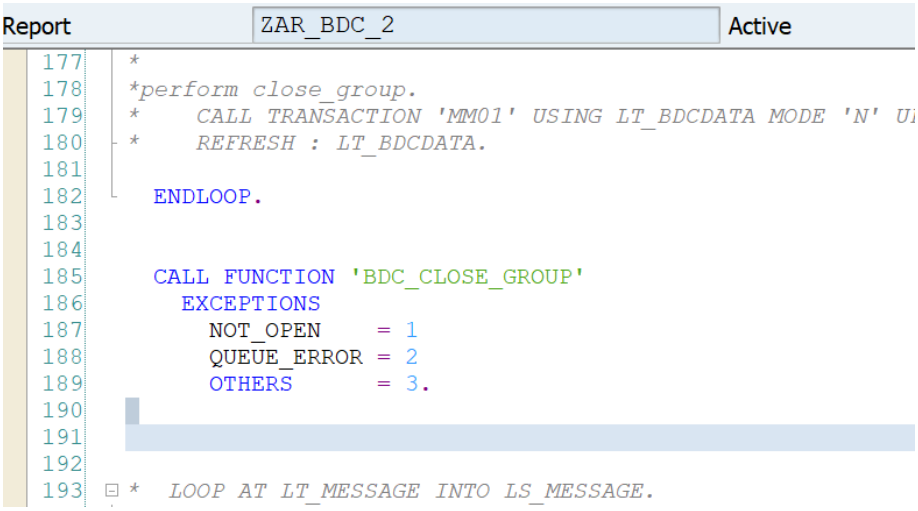
Mob No :- +916261538504

```
  CALL FUNCTION 'BDC_OPEN_GROUP'
   EXPORTING
     CLIENT                          = SY-MANDT
*    DEST                            = FILLER8
     GROUP                           = 'MM01_GRP'
*    HOLDDATE                        = FILLER8
     KEEP                            = 'X'
     USER                            = sy-uname
*    RECORD                          = FILLER1
*    PROG                            = SY-CPROG
*    DCPFM                           = '%'
*    DATFM                           = '%'
* IMPORTING
*    QID                             =
   EXCEPTIONS
     CLIENT_INVALID                  = 1
     DESTINATION_INVALID             = 2
     GROUP_INVALID                   = 3
     GROUP_IS_LOCKED                 = 4
     HOLDDATE_INVALID                = 5
     INTERNAL_ERROR                  = 6
     QUEUE_ERROR                     = 7
     RUNNING                         = 8
     SYSTEM_LOCK_ERROR               = 9
     USER_INVALID                    = 10
     OTHERS                          = 11
           .
```

- Step 2 :- Just after the perform statement we will use the BDC_INSERT function module and will pass MM01 transaction code and internal table of BDCDATA.

```
Report                                                    Active
  PERFORM BDC_FIELD       USING 'MARA-MEINS'
                                LS_MATERIAL-MEINS.



  CALL FUNCTION 'BDC_INSERT'
    EXPORTING
      TCODE            = 'MM01'
*     POST_LOCAL       = NOVBLOCAL
*     PRINTING         = NOPRINT
*     SIMUBATCH        = ' '
*     CTUPARAMS        = ' '
    TABLES
      DYNPROTAB        = LT_BDCDATA
    EXCEPTIONS
      INTERNAL_ERROR   = 1
      NOT_OPEN         = 2
      QUEUE_ERROR      = 3
      TCODE_INVALID    = 4
      PRINTING_INVALID = 5
      POSTING_INVALID  = 6
      OTHERS           = 7.


  REFRESH LT_BDCDATA.
```

- Step 3 :- After the end of loop we will use BDC_CLOSE_GROUP function module to close the group.



Code

```
REPORT ZAR_BDC_01
       NO STANDARD PAGE HEADING LINE-SIZE 255.


TYPES : BEGIN OF TY_MATERIAL,
  MATNR TYPE MATNR,
  MBRSH TYPE MBRSH,
  MTART TYPE MTART,
  MAKTX TYPE MAKTX,
  MEINS TYPE MEINS,
  END OF TY_MATERIAL.


DATA : LT_MATERIAL TYPE TABLE OF TY_MATERIAL,
       LS_MATERIAL TYPE TY_MATERIAL.
DATA : LV_FILE TYPE STRING.
DATA : LT_BDCDATA TYPE TABLE OF BDCDATA,
       LS_BDCDATA TYPE BDCDATA.
```

```
DATA : LT_MESSAGE TYPE TABLE OF BDCMSGCOLL,
       LS_MESSAGE TYPE BDCMSGCOLL.

DATA : LV_MESSAGE TYPE STRING.
PARAMETERS : P_FILE TYPE LOCALFILE.

AT SELECTION-SCREEN ON VALUE-REQUEST FOR P_FILE.
  CALL FUNCTION 'F4_FILENAME'
    EXPORTING
      PROGRAM_NAME  = SYST-CPROG
      DYNPRO_NUMBER = SYST-DYNNR
      FIELD_NAME    = ' '
    IMPORTING
      FILE_NAME     = P_FILE.



START-OF-SELECTION.
  LV_FILE = P_FILE.
  CALL FUNCTION 'GUI_UPLOAD'
    EXPORTING
      FILENAME                = LV_FILE
*     FILETYPE                = 'ASC'
      HAS_FIELD_SEPARATOR     = 'X'
*     HEADER_LENGTH           = 0
*     READ_BY_LINE            = 'X'
*     DAT_MODE                = ' '
*     CODEPAGE                = ' '
*     IGNORE_CERR             = ABAP_TRUE
*     REPLACEMENT             = '#'
*     CHECK_BOM               = ' '
*     VIRUS_SCAN_PROFILE      =
*     NO_AUTH_CHECK           = ' '
*     FILENAME1               = ' '
*   IMPORTING
*     FILELENGTH              =
```

```
*     HEADER                   =
    TABLES
      DATA_TAB                 = LT_MATERIAL
*   CHANGING
*     ISSCANPERFORMED          = ' '
    EXCEPTIONS
      FILE_OPEN_ERROR          = 1
      FILE_READ_ERROR          = 2
      NO_BATCH                 = 3
      GUI_REFUSE_FILETRANSFER = 4
      INVALID_TYPE             = 5
      NO_AUTHORITY             = 6
      UNKNOWN_ERROR            = 7
      BAD_DATA_FORMAT          = 8
      HEADER_NOT_ALLOWED       = 9
      SEPARATOR_NOT_ALLOWED    = 10
      HEADER_TOO_LONG          = 11
      UNKNOWN_DP_ERROR         = 12
      ACCESS_DENIED            = 13
      DP_OUT_OF_MEMORY         = 14
      DISK_FULL                = 15
      DP_TIMEOUT               = 16
      OTHERS                   = 17.



*include bdcrecx1.
*
*start-of-selection.
*


  CALL FUNCTION 'BDC_OPEN_GROUP'
    EXPORTING
      CLIENT              = SY-MANDT
```

```
*     DEST                = FILLER8
      GROUP               = 'MM01_GRP'
*     HOLDDATE            = FILLER8
      KEEP                = 'X'
      USER                = SY-UNAME
*     RECORD              = FILLER1
*     PROG                = SY-CPROG
*     DCPFM               = '%'
*     DATFM               = '%'
* IMPORTING
*     QID                 =
    EXCEPTIONS
      CLIENT_INVALID      = 1
      DESTINATION_INVALID = 2
      GROUP_INVALID       = 3
      GROUP_IS_LOCKED     = 4
      HOLDDATE_INVALID    = 5
      INTERNAL_ERROR      = 6
      QUEUE_ERROR         = 7
      RUNNING             = 8
      SYSTEM_LOCK_ERROR   = 9
      USER_INVALID        = 10
      OTHERS              = 11.
  IF SY-SUBRC <> 0.
* Implement suitable error handling here
  ENDIF.


  LOOP AT LT_MATERIAL INTO LS_MATERIAL.
*perform open_group.

    PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '0060'.
    PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                                  'RMMG1-MTART'.
    PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                                  '=ENTR'.
```

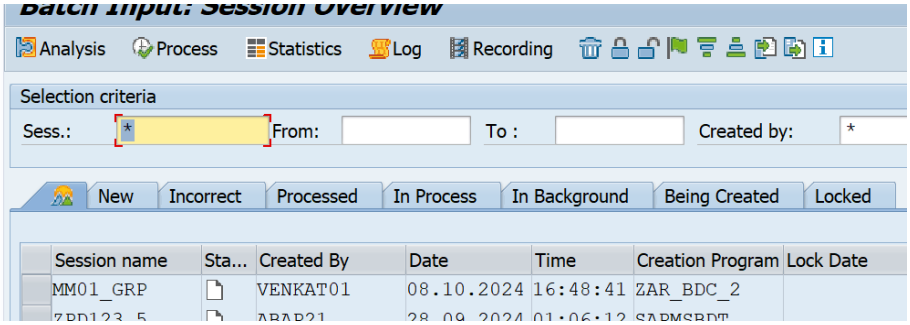
```
    PERFORM BDC_FIELD       USING 'RMMG1-MATNR'
                                  LS_MATERIAL-MATNR.
    PERFORM BDC_FIELD       USING 'RMMG1-MBRSH'
                                  LS_MATERIAL-MBRSH.
    PERFORM BDC_FIELD       USING 'RMMG1-MTART'
                                  LS_MATERIAL-MTART.
    PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '0070'.
    PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                                  'MSICHTAUSW-DYTXT(01)'.
    PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                                  '=ENTR'.
    PERFORM BDC_FIELD       USING 'MSICHTAUSW-KZSEL(01)'
                                  'X'.
    PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '4004'.
    PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                                  '=BU'.
    PERFORM BDC_FIELD       USING 'MAKT-MAKTX'
                                  LS_MATERIAL-MAKTX.
    PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                                  'MARA-MEINS'.
    PERFORM BDC_FIELD       USING 'MARA-MEINS'
                                  LS_MATERIAL-MEINS.



    CALL FUNCTION 'BDC_INSERT'
      EXPORTING
        TCODE            = 'MM01'
*       POST_LOCAL       = NOVBLOCAL
*       PRINTING         = NOPRINT
*       SIMUBATCH        = ' '
*       CTUPARAMS        = ' '
      TABLES
        DYNPROTAB        = LT_BDCDATA
      EXCEPTIONS
        INTERNAL_ERROR   = 1
```

```
        NOT_OPEN         = 2
        QUEUE_ERROR      = 3
        TCODE_INVALID    = 4
        PRINTING_INVALID = 5
        POSTING_INVALID  = 6
        OTHERS           = 7.


    REFRESH LT_BDCDATA.



*perform bdc_transaction using 'MM01'.
*
*perform close_group.
*    CALL TRANSACTION 'MM01' USING LT_BDCDATA MODE 'N' UPDATE
*    REFRESH : LT_BDCDATA.

  ENDLOOP.


  CALL FUNCTION 'BDC_CLOSE_GROUP'
    EXCEPTIONS
      NOT_OPEN    = 1
      QUEUE_ERROR = 2
      OTHERS      = 3.



*  LOOP AT LT_MESSAGE INTO LS_MESSAGE.
*    CALL FUNCTION 'MESSAGE_TEXT_BUILD'
*      EXPORTING
*        MSGID               = LS_MESSAGE-MSGID
*        MSGNR               = LS_MESSAGE-MSGNR
*        MSGV1               = LS_MESSAGE-MSGV1
*        MSGV2               = LS_MESSAGE-MSGV2
```

```
*        MSGV3                = LS_MESSAGE-MSGV3
*        MSGV4                = LS_MESSAGE-MSGV4
*      IMPORTING
*        MESSAGE_TEXT_OUTPUT = LV_MESSAGE.
*
*    WRITE :/ LV_MESSAGE.
*
*  ENDLOOP.



FORM BDC_DYNPRO USING PROGRAM DYNPRO.
  CLEAR LS_BDCDATA.
  LS_BDCDATA-PROGRAM  = PROGRAM.
  LS_BDCDATA-DYNPRO   = DYNPRO.
  LS_BDCDATA-DYNBEGIN = 'X'.
  APPEND LS_BDCDATA TO LT_BDCDATA.
ENDFORM.

*---------------------------------------------------------------------
*        Insert field
*---------------------------------------------------------------------
FORM BDC_FIELD USING FNAM FVAL.
  CLEAR LS_BDCDATA.
  LS_BDCDATA-FNAM = FNAM.
  LS_BDCDATA-FVAL = FVAL.
  APPEND LS_BDCDATA TO LT_BDCDATA.
ENDFORM.
```

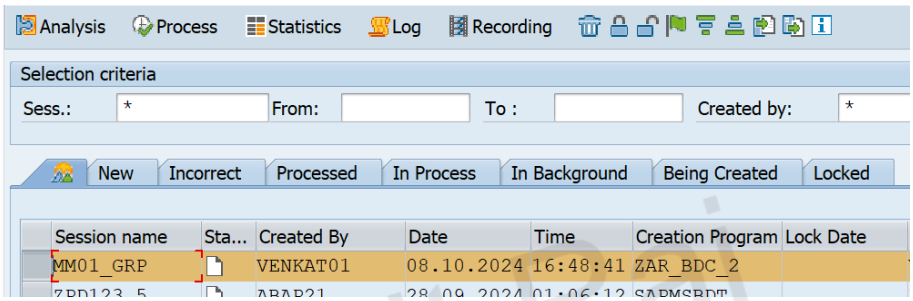

Processing the Session

- After running the program, we will go to SM35 to process the session.
- Step 1 :- Go to SM35 transaction code.

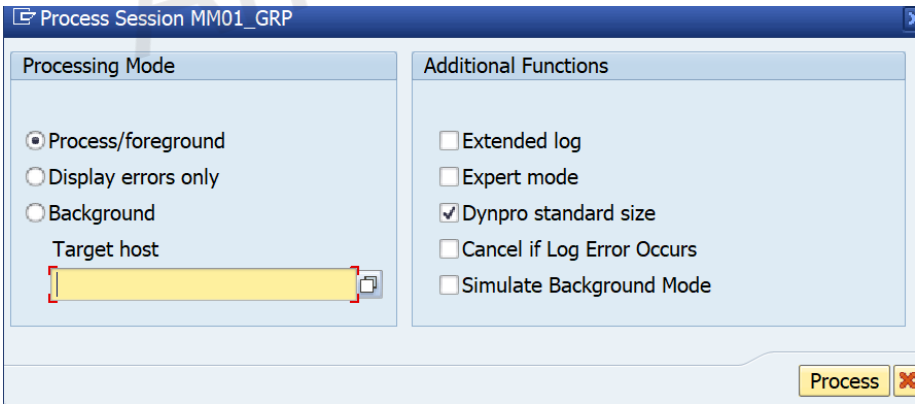


- Here, we can see our session.

- Step 2 :- Select the Session.



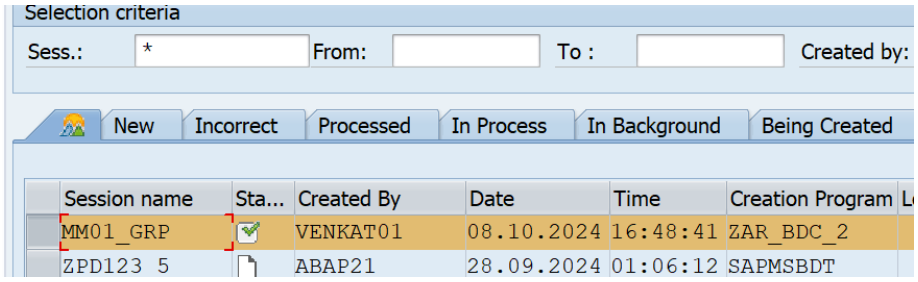
- Click on Process button.



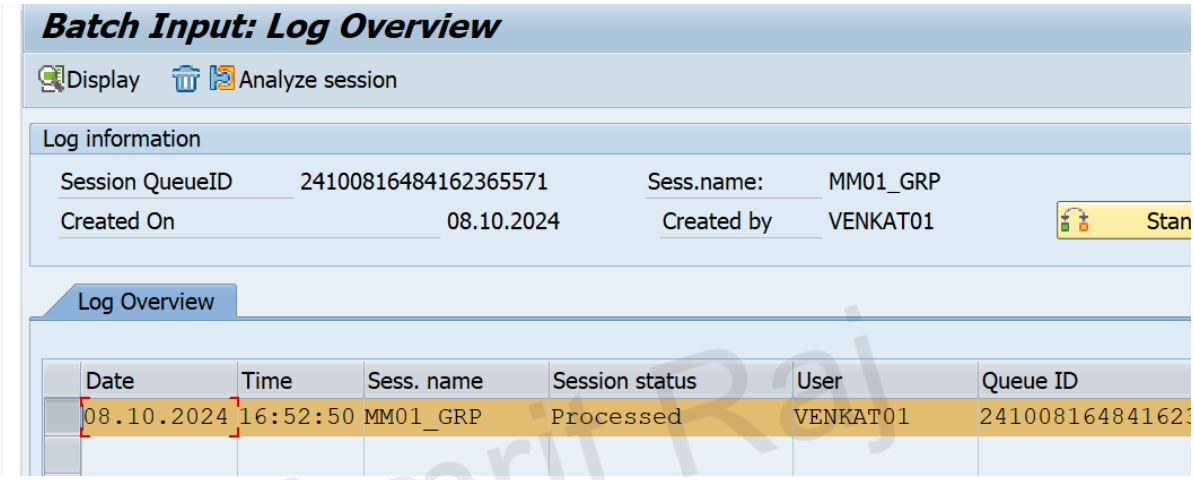
- Step 3 :- If you want to see all the process happening on the screen you can use Process/foreground radio button else, you can use background radio button.

How to See the Log ?

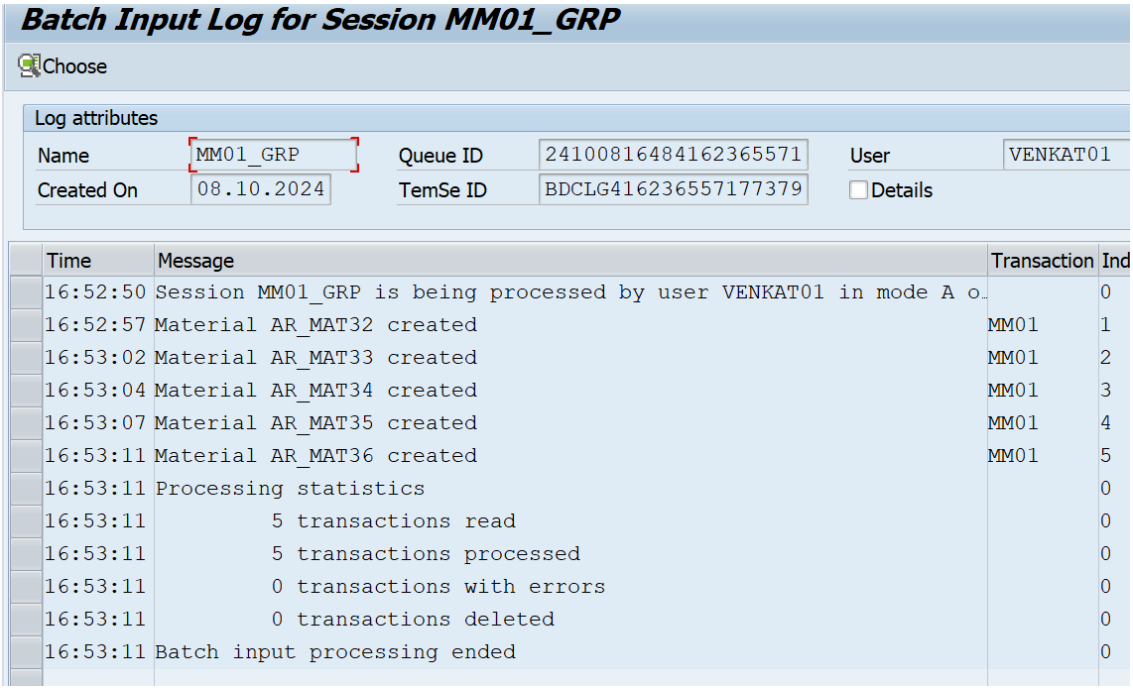
- To see the log after the completion of processing part, so we will select the session.



- Click on log.



- Select the session and click on display button.



- You can see all the materials has been created now.

Comparing Call Transaction and Session Method

- Call transaction statement is used to call the transaction and process the session in CALL Transaction

  where as

  We use Function modules to create and close the group in Session Method.

- The ABAP program does the error handling part using Message tab in Call Transaction

  where as

  In Session Method, ABAP Program creates a session and the transaction SM35 is used to process the session (There is a built in error handling mechanism in this method).

- For Call Transaction, the processing modes are A(All Screen), N(No Screen), E(Error)

  where as

  For Session Method, the Processing Modes are - Foreground, Background, Display Error Only.

- For Call Transaction, the database update modes are Synchronous, Asynchronous and Local Update.

  where as

  In Session method Only synchronous database update

- For Call Transaction, there is no batch input processing log

  where as

  For Session Method, There is a detailed log which is generated for all sessions.

- Overall, Call Transaction method is faster than Session Method.

4️⃣

BAPI

1. Introduction

- BAPI stands for Business Application Programming Interface
- BAPI is a technique which is used to conduct a large scale data migration
- BAPIs in the SAP Systems are implemented as RFC enabled function modules.
- When we create a business object for a RFC enabled function module and we merge it with the function module then we call it a BAPI.
    - i.e. BAPI = RFC Function Module + Business Object
- BAPI is always preferable as compared to BDC for data migration.

---

2. Why BAPI is more preferable as compared to BDC ?

1. BDC

- Upgradation :- When we go for upgradation, there might be possibility to change the screen elements or numbers depending on the requirement. In that case, our existing BDC Program

may or may not work. Unless and util we prepare new BDC recording, So, in that case we can’t use the OLD BDC Program.

- Faster :- BDC run through the screen flow to update the database. So, It is slower as compared to BAPI.

2. BAPI

- Upgradation :- In BAPI, SAP promises that they are going to keep the old BAPI and for new functionality they will provide an upgraded BAPI. Until we write a new BAPI Program, we can use the existing BAPI Program.
- Faster :- BAPI directly updates the database, So it is faster as compared to BDC.

---

3. Some Important BAPIs for Data Migration

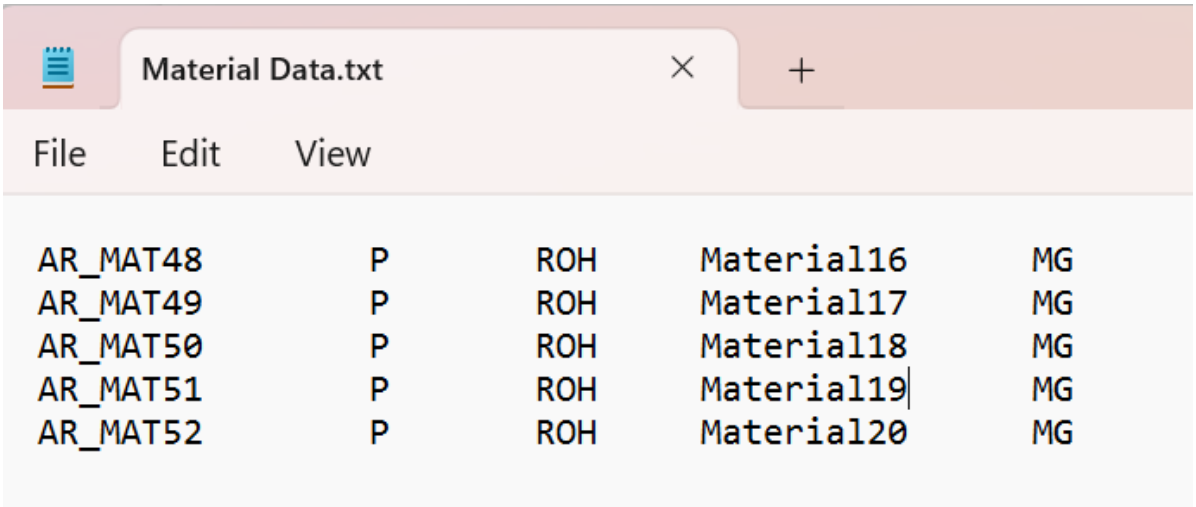
- Create and Change Material Master Data :-
  - BAPI_MATERIAL_SAVEDATA
- Creating a Purchase Order :-
  - BAPI_PO_CREATE1

---

4. Requirement

- We will upload the below material data from notepad using BAPI.

Mob No :- +916261538504



5. Implementation

- Step 1 :- We will create a type structure and internal table for the same.

```
TYPES : BEGIN OF TY_MATERIAL,
   MATNR TYPE MATNR,
   MBRSH TYPE MBRSH,
   MTART TYPE MTART,
   MAKTX TYPE MAKTX,
   MEINS TYPE MEINS,
   END OF TY_MATERIAL.


DATA : LT_MATERIAL TYPE TABLE OF TY_MATERIAL,
       LS_MATERIAL TYPE TY_MATERIAL.
DATA : LV FILE TYPE STRING.
```

- Step 2 :- We will use At Selection Screen on value request event and F4_Filenname method to pick our file from local system.

```
PARAMETERS : P_FILE TYPE LOCALFILE.

AT SELECTION-SCREEN ON VALUE-REQUEST FOR P_FILE.
  CALL FUNCTION 'F4_FILENAME'
    EXPORTING
      PROGRAM_NAME  = SYST-CPROG
      DYNPRO_NUMBER = SYST-DYNNR
      FIELD_NAME    = ' '
    IMPORTING
      FILE_NAME     = P_FILE.
```


Mob No :- +916261538504

- Step 3 :- Then In Start of Selection Event we will upload our File using GUI_UPLOAD method.

```
START-OF-SELECTION.
  LV_FILE = P_FILE.
  CALL FUNCTION 'GUI_UPLOAD'
    EXPORTING
      FILENAME                = LV_FILE
*     FILETYPE                = 'ASC'
      HAS_FIELD_SEPARATOR     = 'X'
*     HEADER_LENGTH           = 0
*     READ_BY_LINE            = 'X'
*     DAT_MODE                = ' '
*     CODEPAGE                = ' '
*     IGNORE_CERR             = ABAP_TRUE
*     REPLACEMENT             = '#'
*     CHECK_BOM               = ' '
*     VIRUS_SCAN_PROFILE      =
*     NO_AUTH_CHECK           = ' '
*     FILENAME1               = ' '
*   IMPORTING
*     FILELENGTH              =
*     HEADER                  =
    TABLES
      DATA_TAB                = LT_MATERIAL
*   CHANGING
*     ISSCANPERFORMED         = ' '
    EXCEPTIONS
      FILE_OPEN_ERROR         = 1
      FILE_READ_ERROR         = 2
      NO_BATCH                = 3
      GUI_REFUSE_FILETRANSFER = 4
      INVALID_TYPE            = 5
      NO_AUTHORITY            = 6
      UNKNOWN_ERROR           = 7
      BAD_DATA_FORMAT         = 8
      HEADER_NOT_ALLOWED      = 9
      SEPARATOR_NOT_ALLOWED   = 10
```


- Step 4 :- Then we will loop on our internal table.

```
LOOP at LT_MATERIAL into LS_MATERIAL.



ENDLOOP.
```

- Step 5 :- Inside the loop we will call the BAPI_MATERIAL_SAVEDATA function module.

Mob No :- +916261538504

```
82  LOOP at LT_MATERIAL into LS_MATERIAL.
83
84    CALL FUNCTION 'BAPI_MATERIAL_SAVEDATA'
85      EXPORTING
86        HEADDATA                =
87  *     CLIENTDATA              =
88  *     CLIENTDATAX             =
89  *     PLANTDATA               =
90  *     PLANTDATAX              =
91  *     FORECASTPARAMETERS      =
92  *     FORECASTPARAMETERSX     =
93  *     PLANNINGDATA            =
94  *     PLANNINGDATAX           =
95  *     STORAGELOCATIONDATA     =
96  *     STORAGELOCATIONDATAX    =
97  *     VALUATIONDATA           =
98  *     VALUATIONDATAX          =
99  *     WAREHOUSENUMBERDATA     =
100 *     WAREHOUSENUMBERDATAX    =
101 *     SALESDATA               =
102 *     SALESDATAX              =
103 *     STORAGETYPEDATA         =
104 *     STORAGETYPEDATAX        =
105 *     FLAG_ONLINE             = ' '
```

- Step 6 :- For the HEADATA parameter, we will create a work area and in that work area we will pass MATERIAL, IND_SECTOR, MATL_TYPE, BASIC_VIEW.

```
23
24  DATA : LS_HEADDATA TYPE BAPIMATHEAD.
25
26
```

```
83
84    LOOP AT LT_MATERIAL INTO LS_MATERIAL.
85
86      LS_HEADDATA-MATERIAL = LS_MATERIAL-MATNR.
87      LS_HEADDATA-IND_SECTOR = LS_MATERIAL-MBRSH.
88      LS_HEADDATA-MATL_TYPE = LS_MATERIAL-MTART.
89      LS_HEADDATA-BASIC_VIEW = 'X'.
90
91      CALL FUNCTION 'BAPI_MATERIAL_SAVEDATA'
92        EXPORTING
93          HEADDATA = LS_HEADDATA
94  *       CLIENTDATA              =
95  *       CLIENTDATAX             =
96  *       PLANTDATA               =
97  *       PLANTDATAX              =
```

- Step 7 :- We will create a work area for CLIENTDATA and we will pass unit of measurement into it.

```
4   DATA : LS_HEADDATA TYPE BAPIMATHEAD.
5   DATA : LS_CLIENTDATA TYPE BAPI_MARA.
6
```

```
89      LS_HEADDATA-BASIC_VIEW = 'X'.
90
91      LS_CLIENTDATA-BASE_UOM = LS_MATERIAL-MEINS.
92
93      CALL FUNCTION 'BAPI_MATERIAL_SAVEDATA'
94        EXPORTING
95          HEADDATA    = LS_HEADDATA
96          CLIENTDATA = LS_CLIENTDATA
97  *       CLIENTDATAX                =
98  *       PLANTDATA  =
99  *       PLANTDATAX =
```

- Step 8 :- Whatever data you will pass in CLIENTATA, you will have to mark it as TRUE in CLIENTDATAX parameter.

```
24  DATA : LS_HEADDATA TYPE BAPIMATHEAD.
25  DATA : LS_CLIENTDATA TYPE BAPI_MARA.
26  DATA : LS_CLIENTDATAX TYPE  BAPI_MARAX.
27
28  PARAMETERS : P_FILE TYPE LOCALFILE.
```

```
91
92      LS_CLIENTDATA-BASE_UOM = LS_MATERIAL-MEINS.
93      LS_CLIENTDATAX-BASE_UOM = 'X'.
94
95      CALL FUNCTION 'BAPI_MATERIAL_SAVEDATA'
96        EXPORTING
97          HEADDATA    = LS_HEADDATA
98          CLIENTDATA  = LS_CLIENTDATA
99          CLIENTDATAX = LS_CLIENTDATAX
100 *       PLANTDATA   =
101 *       PLANTDATAX  =
102 *       FORECASTPARAMETERS         =
```

- Step 9 :- Now we will create a internal table for MATERIALDESCRIPTION parameter and we will pass our material description and language in this table.

```
26  DATA : LS_CLIENTDATAX TYPE  BAPI_MARAX.
27
28  DATA : LT_DESCRIPTION TYPE TABLE OF BAPI_MAKT,
29         LS_DESCRIPTION TYPE BAPI_MAKT.
30
31  PARAMETERS : P_FILE TYPE LOCALFILE.
```


Mob No :- +916261538504

```
94
95      LS_CLIENTDATA-BASE_UOM = LS_MATERIAL-MEINS.
96      LS_CLIENTDATAX-BASE_UOM = 'X'.
97
98      LS_DESCRIPTION-LANGU = SY-LANGU.
99      LS_DESCRIPTION-MATL_DESC = LS_MATERIAL-MAKTX.
100     APPEND LS_DESCRIPTION TO LT_DESCRIPTION.
101
102     CALL FUNCTION 'BAPI_MATERIAL_SAVEDATA'
```

and we will pass this into MATERIALDESCRIPTION parameter.

```
26  *       NO_ROLLBACK_WORK     = ' '
27  *   IMPORTING
28  *       RETURN               =
29        TABLES
30          MATERIALDESCRIPTION = LT_DESCRIPTION
31  *       UNITSOFMEASURE       =
32  *       UNITSOFMEASUREX      =
```

- Step 10 :- We will create a return work area for the RETURN parameter.

```
30
31    DATA : LT_RETURN TYPE TABLE OF BAPIRET2,
32           LS_RETURN TYPE BAPIRET2.
33
34    PARAMETERS : P_FILE TYPE LOCALFILE.
```

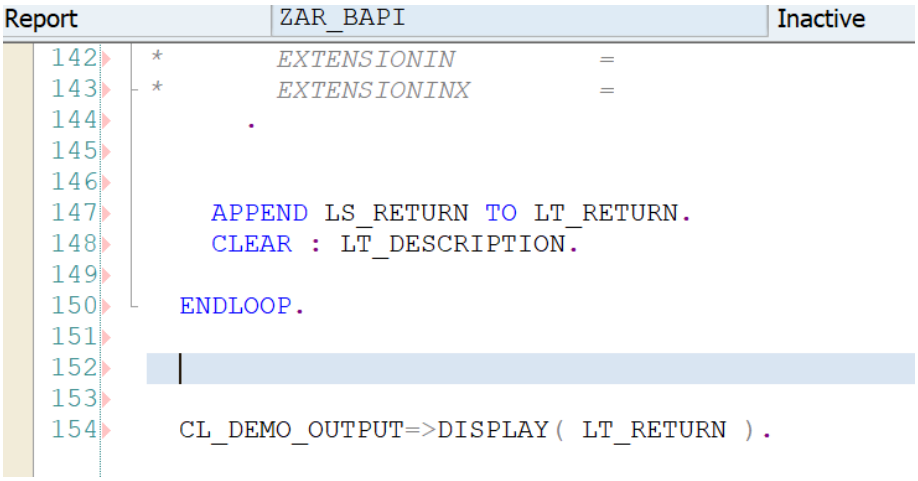
then we will pass the work area into RETURN parameter.

```
127 *       FLAG_CAD_CALL        = ' '
128 *       NO_DEQUEUE           = ' '
129 *       NO_ROLLBACK_WORK     = ' '
130       IMPORTING
131         RETURN               = LS_RETURN
132       TABLES
133         MATERIALDESCRIPTION = LT_DESCRIPTION
```

- Step 10 :- Just before ending of loop we will append the RETURN work area into internal table and we will clear the description internal table.

```
145
146
147     APPEND LS_RETURN TO LT_RETURN.
148     CLEAR : LT_DESCRIPTION.
149
150   ENDLOOP.
```

- Step 11 :- After the ending of loop we will use DISPLAY our return table.



6. Code

```
TYPES : BEGIN OF TY_MATERIAL,
  MATNR TYPE MATNR,
  MBRSH TYPE MBRSH,
  MTART TYPE MTART,
  MAKTX TYPE MAKTX,
  MEINS TYPE MEINS,
  END OF TY_MATERIAL.


DATA : LT_MATERIAL TYPE TABLE OF TY_MATERIAL,
       LS_MATERIAL TYPE TY_MATERIAL.
DATA : LV_FILE TYPE STRING.

DATA : LS_HEADDATA TYPE BAPIMATHEAD.
DATA : LS_CLIENTDATA TYPE BAPI_MARA.
DATA : LS_CLIENTDATAX TYPE  BAPI_MARAX.

DATA : LT_DESCRIPTION TYPE TABLE OF BAPI_MAKT,
       LS_DESCRIPTION TYPE BAPI_MAKT.

DATA : LT_RETURN TYPE TABLE OF BAPIRET2,
```

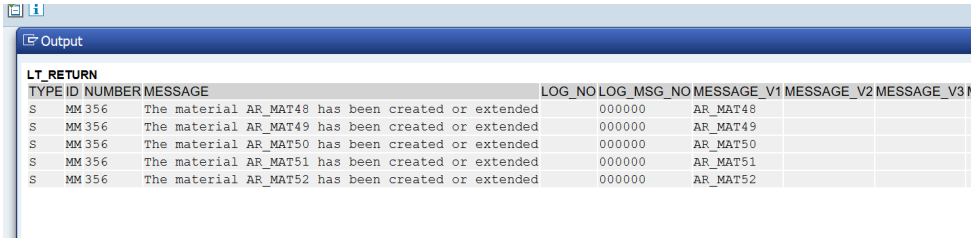
```
       LS_RETURN TYPE BAPIRET2.

PARAMETERS : P_FILE TYPE LOCALFILE.

AT SELECTION-SCREEN ON VALUE-REQUEST FOR P_FILE.
  CALL FUNCTION 'F4_FILENAME'
    EXPORTING
      PROGRAM_NAME  = SYST-CPROG
      DYNPRO_NUMBER = SYST-DYNNR
      FIELD_NAME    = ' '
    IMPORTING
      FILE_NAME     = P_FILE.



START-OF-SELECTION.
  LV_FILE = P_FILE.
  CALL FUNCTION 'GUI_UPLOAD'
    EXPORTING
      FILENAME                = LV_FILE
*     FILETYPE                = 'ASC'
      HAS_FIELD_SEPARATOR     = 'X'
*     HEADER_LENGTH           = 0
*     READ_BY_LINE            = 'X'
*     DAT_MODE                = ' '
*     CODEPAGE                = ' '
*     IGNORE_CERR             = ABAP_TRUE
*     REPLACEMENT             = '#'
*     CHECK_BOM               = ' '
*     VIRUS_SCAN_PROFILE      =
*     NO_AUTH_CHECK           = ' '
*     FILENAME1               = ' '
*   IMPORTING
*     FILELENGTH              =
*     HEADER                  =
    TABLES
```

```
      DATA_TAB                = LT_MATERIAL
*   CHANGING
*     ISSCANPERFORMED         = ' '
    EXCEPTIONS
      FILE_OPEN_ERROR         = 1
      FILE_READ_ERROR         = 2
      NO_BATCH                = 3
      GUI_REFUSE_FILETRANSFER = 4
      INVALID_TYPE            = 5
      NO_AUTHORITY            = 6
      UNKNOWN_ERROR           = 7
      BAD_DATA_FORMAT         = 8
      HEADER_NOT_ALLOWED      = 9
      SEPARATOR_NOT_ALLOWED   = 10
      HEADER_TOO_LONG         = 11
      UNKNOWN_DP_ERROR        = 12
      ACCESS_DENIED           = 13
      DP_OUT_OF_MEMORY        = 14
      DISK_FULL               = 15
      DP_TIMEOUT              = 16
      OTHERS                  = 17.


  LOOP AT LT_MATERIAL INTO LS_MATERIAL.

    LS_HEADDATA-MATERIAL = LS_MATERIAL-MATNR.
    LS_HEADDATA-IND_SECTOR = LS_MATERIAL-MBRSH.
    LS_HEADDATA-MATL_TYPE = LS_MATERIAL-MTART.
    LS_HEADDATA-BASIC_VIEW = 'X'.

    LS_CLIENTDATA-BASE_UOM = LS_MATERIAL-MEINS.
    LS_CLIENTDATAX-BASE_UOM = 'X'.

    LS_DESCRIPTION-LANGU = SY-LANGU.
    LS_DESCRIPTION-MATL_DESC = LS_MATERIAL-MAKTX.
    APPEND LS_DESCRIPTION TO LT_DESCRIPTION.
```

```
    CLEAR LS_DESCRIPTION.

    CALL FUNCTION 'BAPI_MATERIAL_SAVEDATA'
      EXPORTING
        HEADDATA            = LS_HEADDATA
        CLIENTDATA          = LS_CLIENTDATA
        CLIENTDATAX         = LS_CLIENTDATAX
*       PLANTDATA           =
*       PLANTDATAX          =
*       FORECASTPARAMETERS  =
*       FORECASTPARAMETERSX =
*       PLANNINGDATA        =
*       PLANNINGDATAX       =
*       STORAGELOCATIONDATA =
*       STORAGELOCATIONDATAX       =
*       VALUATIONDATA       =
*       VALUATIONDATAX      =
*       WAREHOUSENUMBERDATA =
*       WAREHOUSENUMBERDATAX   =
*       SALESDATA           =
*       SALESDATAX          =
*       STORAGETYPEDATA     =
*       STORAGETYPEDATAX    =
*       FLAG_ONLINE         = ' '
*       FLAG_CAD_CALL       = ' '
*       NO_DEQUEUE          = ' '
*       NO_ROLLBACK_WORK    = ' '
      IMPORTING
        RETURN              = LS_RETURN
      TABLES
        MATERIALDESCRIPTION = LT_DESCRIPTION
*       UNITSOFMEASURE      =
*       UNITSOFMEASUREX     =
*       INTERNATIONALARTNOS =
*       MATERIALLONGTEXT    =
*       TAXCLASSIFICATIONS  =
```

```
*        RETURNMESSAGES       =
*        PRTDATA              =
*        PRTDATAX             =
*        EXTENSIONIN          =
*        EXTENSIONINX         =
       .


    APPEND LS_RETURN TO LT_RETURN.
    CLEAR : LT_DESCRIPTION.

  ENDLOOP.



  CL_DEMO_OUTPUT=>DISPLAY( LT_RETURN ).
```


7. Output



Output

LT_RETURN

TYPE	ID	NUMBER	MESSAGE	LOG_NO	LOG_MSG_NO	MESSAGE_V1	MESSAGE_V2	MESSAGE_V3
S	MM	356	The material AR_MAT48 has been created or extended		000000	AR_MAT48		
S	MM	356	The material AR_MAT49 has been created or extended		000000	AR_MAT49		
S	MM	356	The material AR_MAT50 has been created or extended		000000	AR_MAT50		
S	MM	356	The material AR_MAT51 has been created or extended		000000	AR_MAT51		
S	MM	356	The material AR_MAT52 has been created or extended		000000	AR_MAT52		

8. Checking data in MARA TABLE

MANDT	MATNR	ERSDA	ERNAM	LAEDA	AENAM	VPSTA	PSTAT	LVORM	MTART	MBRSH	MATKL
800	AR_MAT48	09.10.2024	VENKAT01	00.00.0000		K	K		ROH	P	
800	AR_MAT49	09.10.2024	VENKAT01	00.00.0000		K	K		ROH	P	
800	AR_MAT50	09.10.2024	VENKAT01	00.00.0000		K	K		ROH	P	
800	AR_MAT51	09.10.2024	VENKAT01	00.00.0000		K	K		ROH	P	
800	AR_MAT52	09.10.2024	VENKAT01	00.00.0000		K	K		ROH	P	