Objective:

- **Batch Data Communication**
- **Types of BDC and differences between them**
- **File Handling in SAP both on application server and presentation server.**

Definition :

Batch Data Communication (BDC) is the process of transferring data from one SAP System to another SAP system or from a non-SAP system to SAP System.

Features :

- **BDC is an automatic procedure.**
- **This method is used to transfer large amount of data that is available in electronic medium.**
- **BDC can be used primarily when installing the SAP system and when transferring data from a legacy system (external system).**
- **BDC uses normal transaction codes to transfer data.**

Types of BDC :

- **CLASSICAL BATCH INPUT (Session Method)**
- **CALL TRANSACTION**

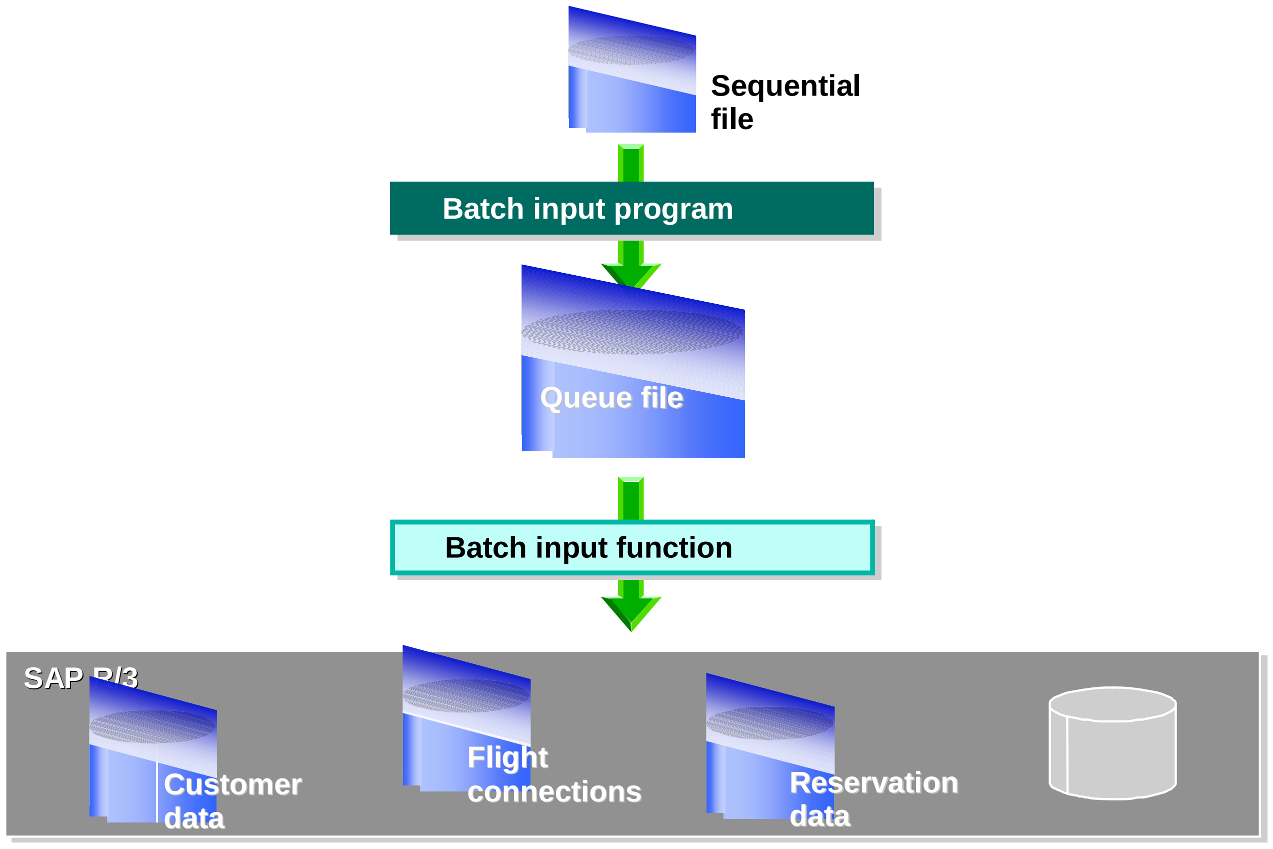
BATCH INPUT METHOD:

This method is also called as ‘CLASSICAL METHOD’.

Features:

- **Asynchronous processing.**
- **Synchronous Processing in database update**
- **Transfer data for more than one transaction.**
- **Batch input processing log will be generated.**
- **During processing, no transaction is started until the previous transaction has been written to the database.**

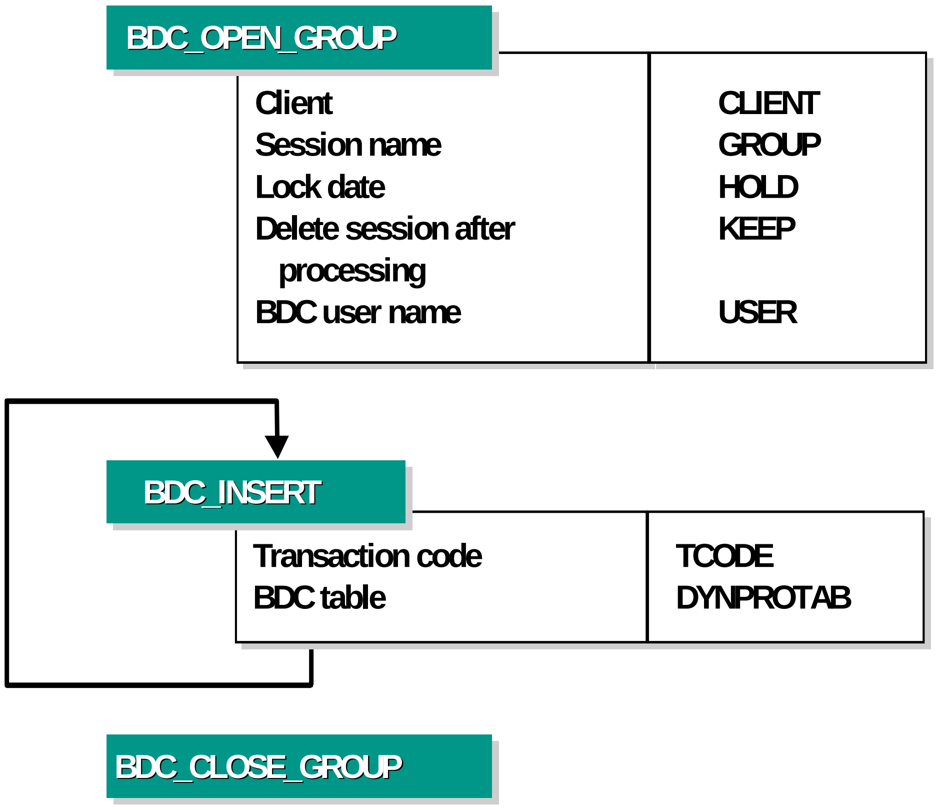
Batch Input Processing



Steps involved in Classical Batch Input :

- **Analyze the data that is to be transferred to the SAP system to determine how the existing data should be mapped to the SAP data structure.**
- **Generate SAP data structures for incorporation into the data export program (SAP provides one method called Recording to generate the SAP data structure and transaction code for this is SHDB).**
- **Read data in, often from a sequential file that has been exported from another system or prepared by a data transfer program.**
- **Perform data conversions, if necessary.**
- **Prepare the data for batch input processing by storing the data in the batch input data structure, BDCDATA.**
- **Generate a batch input session using the function modules BDC_OPEN_GROUP, BDC_INSERT and BDC_CLOSE_GROUP (the parameters to these function modules explained in the next slide).**
- **Process the session from System → Services → Batch Input (Transaction code is SM35).**

Function Modules & Parameters for Session Method



BDC_OPEN_GROUP

Client	CLIENT
Session name	GROUP
Lock date	HOLD
Delete session after processing	KEEP
BDC user name	USER

BDC_INSERT

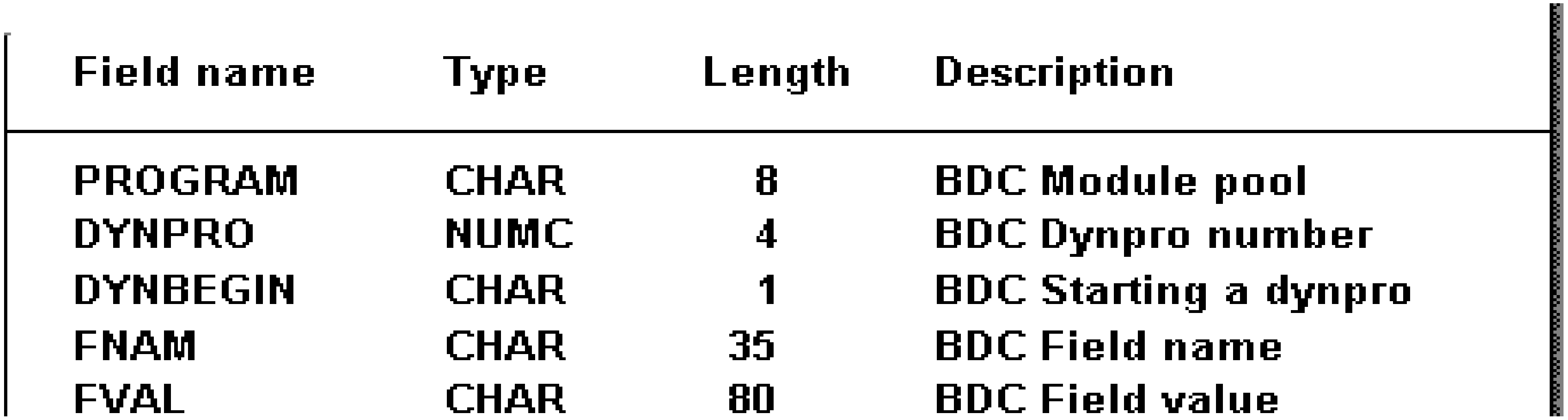
Transaction code	TCODE
BDC table	DYNPROTAB

BDC_CLOSE_GROUP

Structure of the BDCDATA:

BDC Table

Program	Screen	Start	Field name	Field contents
<program name>	<number 1>	x		
			<field 11>	<value 11>
			<field 12>	<value 12>
			...	...
<program name>	<number 2>	x		
			<field 21>	<value 21>
			<field 22>	<value 22>
			...	...
			...	...



Field name	Type	Length	Description
PROGRAM	CHAR	8	BDC Module pool
DYNPRO	NUMC	4	BDC Dynpro number
DYNBEGIN	CHAR	1	BDC Starting a dynpro
FNAM	CHAR	35	BDC Field name
FVAL	CHAR	80	BDC Field value

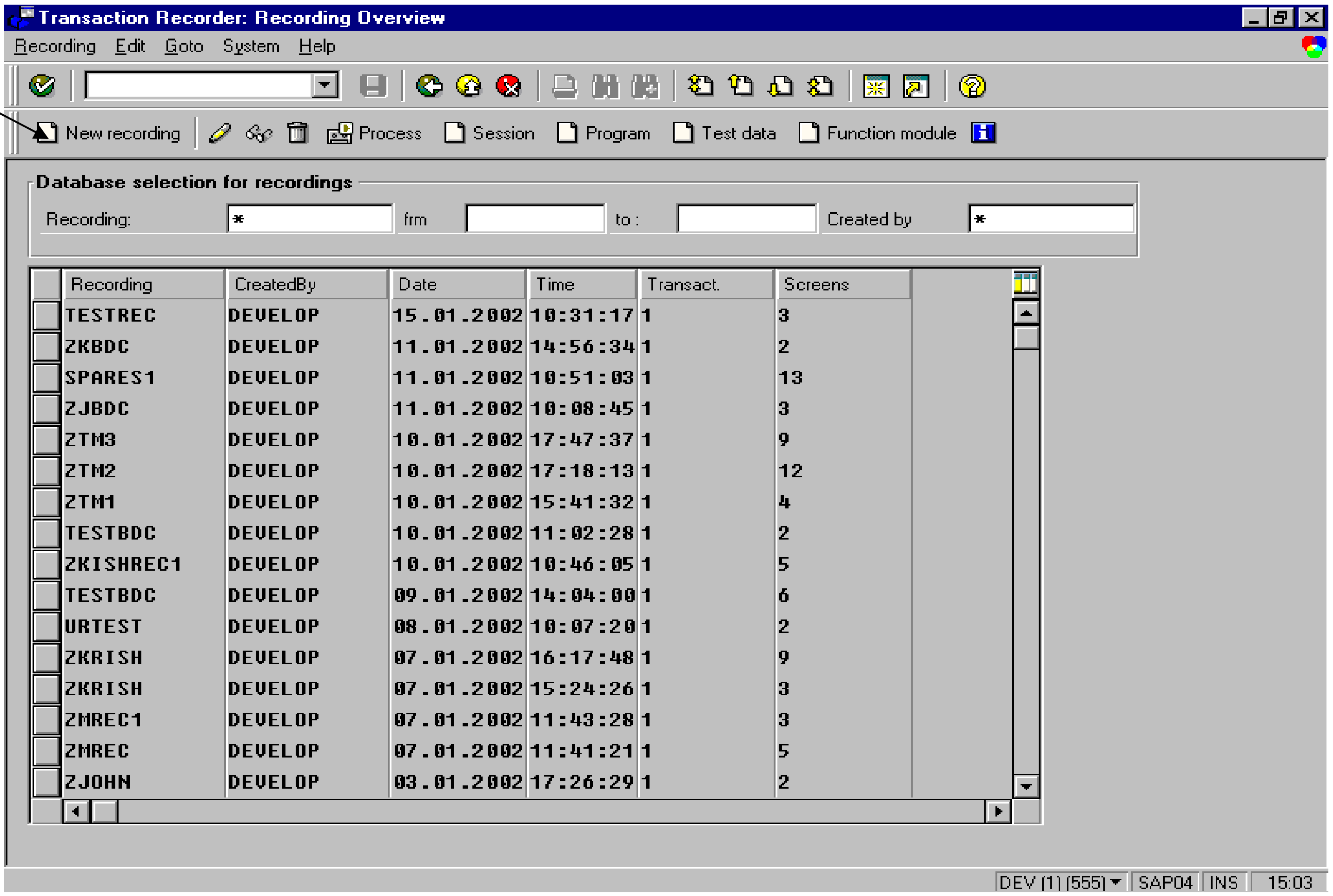
Recording (Transaction code SHDB)

Recording is a process that is provided by the SAP system to generate the SAP data structure for batch data communication.

A sample recording for the transaction MK01 is explained below.

Recording Step 1:
Starting screen of the recording (Transaction code SHDB)

Click “New Recording” button to new recording.


Transaction Recorder: Recording Overview
Recording Edit Goto System Help
New recording
Process
Session
Program
Test data
Function module

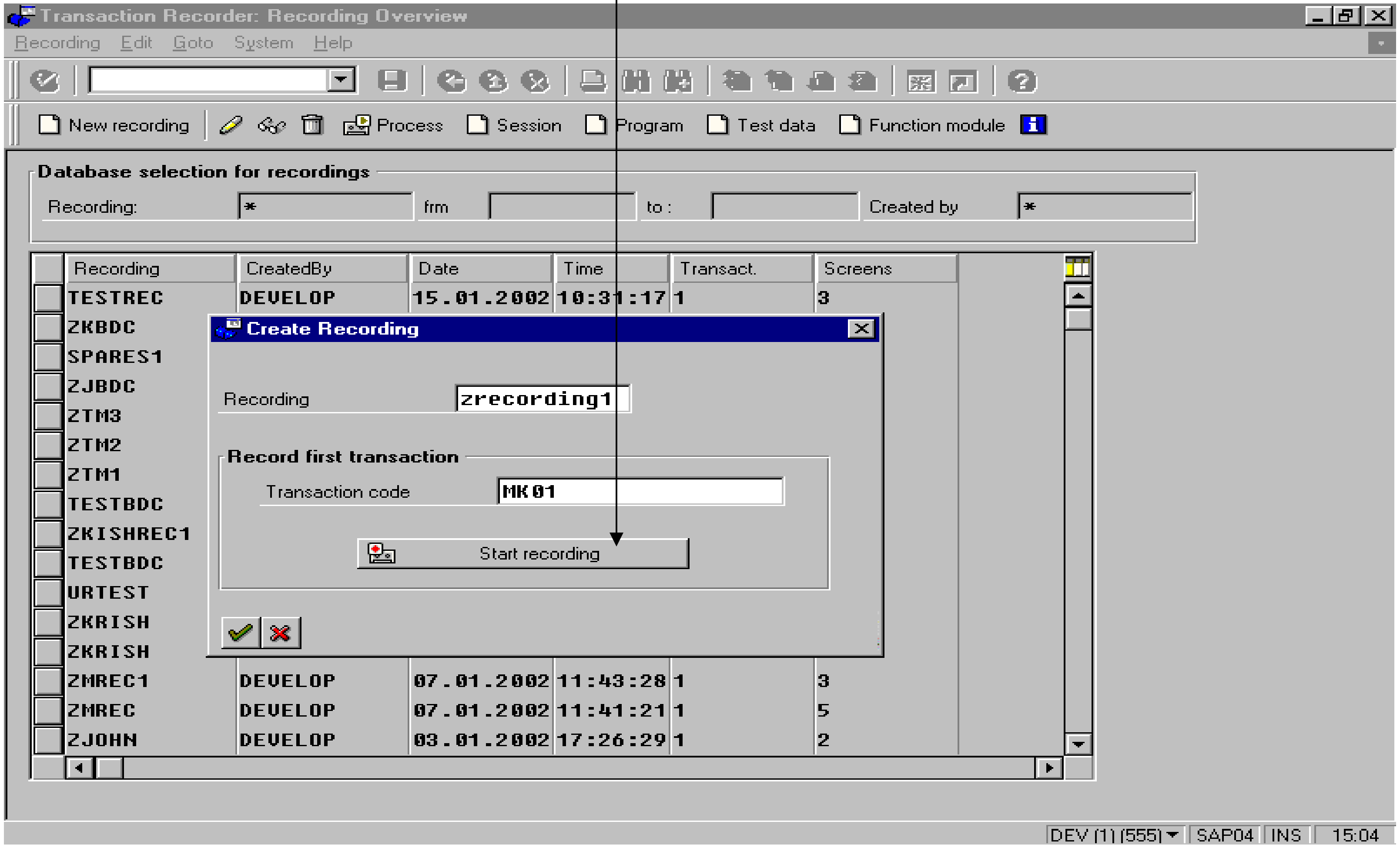
Database selection for recordings

Recording:	*	frm		to :		Created by	*

Recording	CreatedBy	Date	Time	Transact.	Screens
TESTREC	DEVELOP	15.01.2002	10:31:17	1	3
ZKBDC	DEVELOP	11.01.2002	14:56:34	1	2
SPARES1	DEVELOP	11.01.2002	10:51:03	1	13
ZJBDC	DEVELOP	11.01.2002	10:08:45	1	3
ZTM3	DEVELOP	10.01.2002	17:47:37	1	9
ZTM2	DEVELOP	10.01.2002	17:18:13	1	12
ZTM1	DEVELOP	10.01.2002	15:41:32	1	4
TESTBDC	DEVELOP	10.01.2002	11:02:28	1	2
ZKISHREC1	DEVELOP	10.01.2002	10:46:05	1	5
TESTBDC	DEVELOP	09.01.2002	14:04:00	1	6
URTEST	DEVELOP	08.01.2002	10:07:20	1	2
ZKRISH	DEVELOP	07.01.2002	16:17:48	1	9
ZKRISH	DEVELOP	07.01.2002	15:24:26	1	3
ZMREC1	DEVELOP	07.01.2002	11:43:28	1	3
ZMREC	DEVELOP	07.01.2002	11:41:21	1	5
ZJOHN	DEVELOP	03.01.2002	17:26:29	1	2

DEV (1) (555) SAP04 INS 15:03

Recording Step 2: Enter the recording name and the transaction for which you want the recording and click “Start Recording” Button.



Database selection for recordings

Recording: * frm to : Created by *

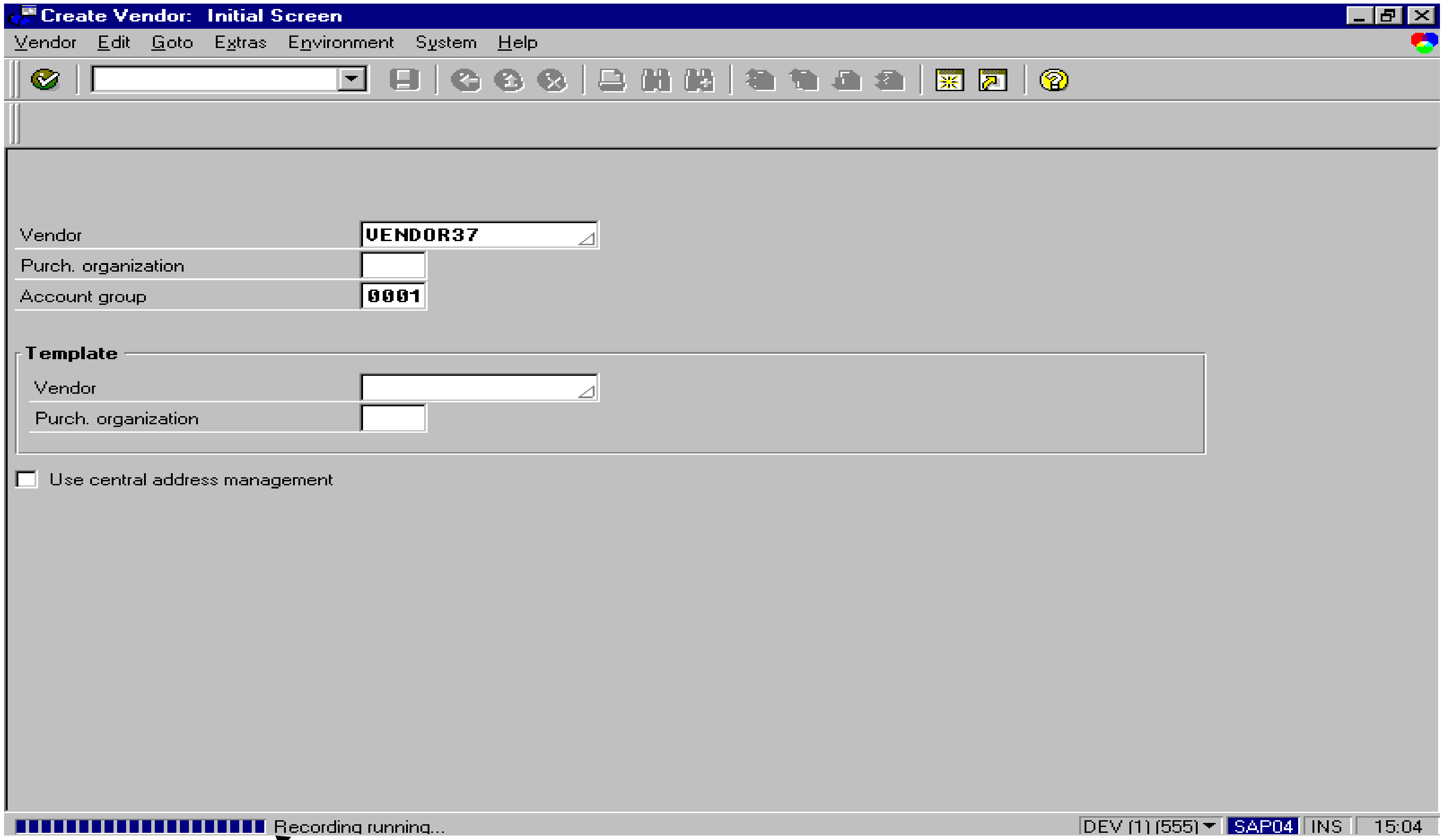
Recording	CreatedBy	Date	Time	Transact.	Screens
TESTREC	DEVELOP	15.01.2002	10:31:17	1	3
ZKBDC					
SPARES1					
ZJBDC					
ZTM3					
ZTM2					
ZTM1					
TESTBDC					
ZKISHREC1					
TESTBDC					
URTEST					
ZKRISH					
ZKRISH					
ZMREC1	DEVELOP	07.01.2002	11:43:28	1	3
ZMREC	DEVELOP	07.01.2002	11:41:21	1	5
ZJOHN	DEVELOP	03.01.2002	17:26:29	1	2

Recording zrecording1

Record first transaction

Transaction code MK01

Recording Step 3: Enter sample data for the transaction.


Create Vendor: Initial Screen
Vendor Edit Goto Extras Environment System Help

Vendor	VENDOR37
Purch. organization	
Account group	0001

Template

Vendor	
Purch. organization	

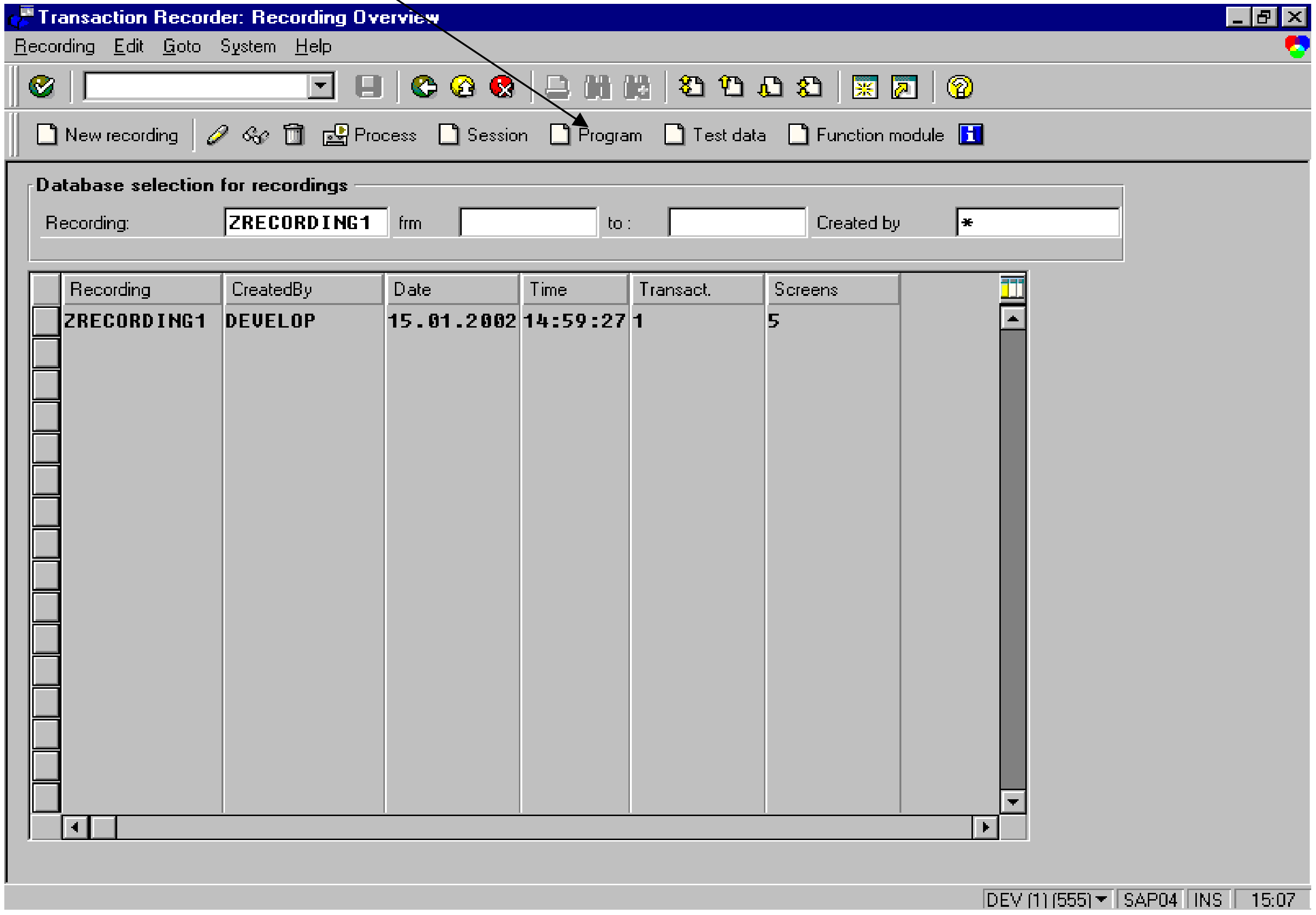
☐ Use central address management

Recording running...

DEV (1) (555) SAP04 INS 15:04

Every key stroke will be recorded.

Recording Step 4: After entering all data for the transaction (here MK01), the recording overview screen will be displayed. Then save the recording. And click the “Program” button.

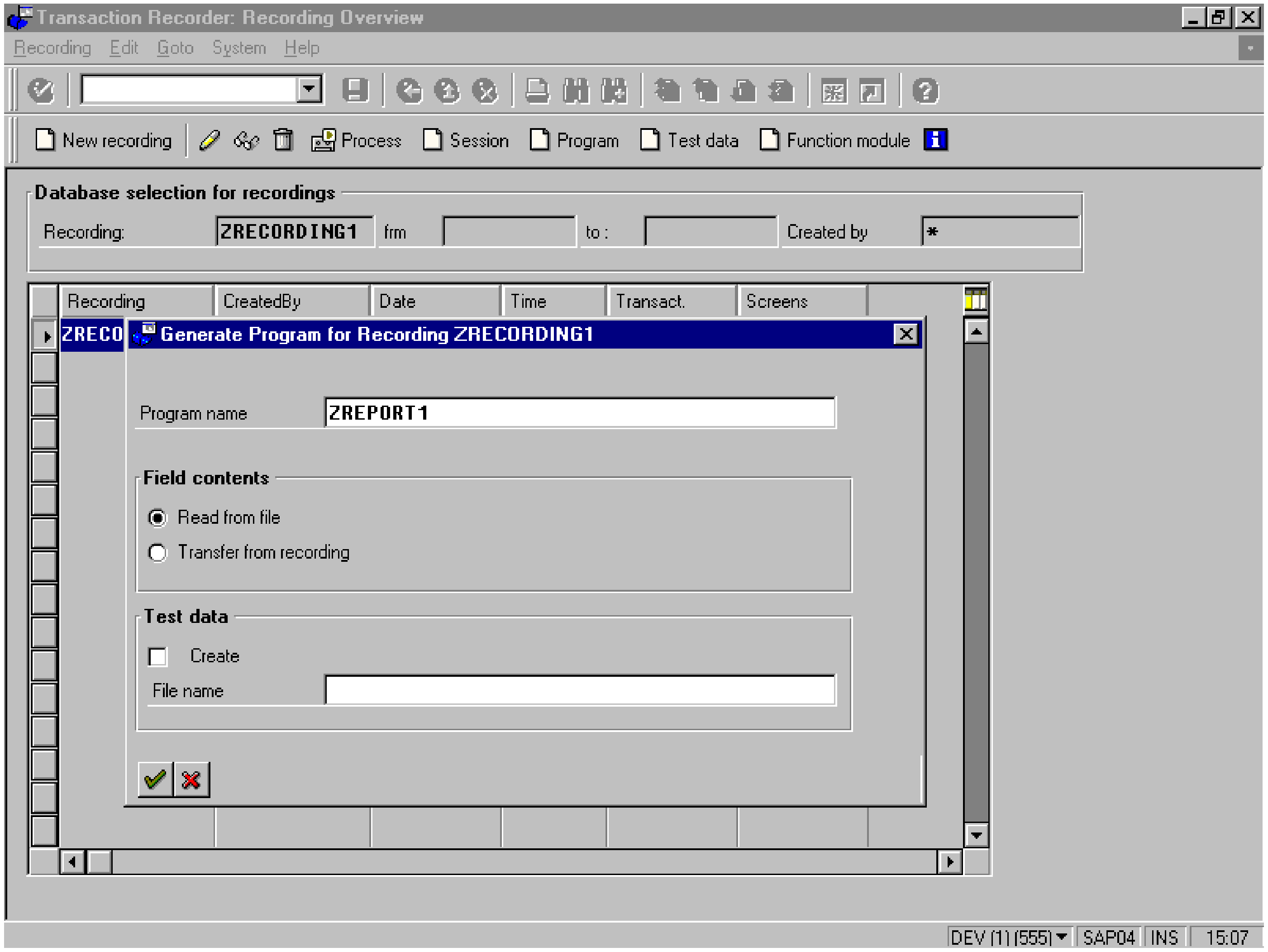

Transaction Recorder: Recording Overview
Recording Edit Goto System Help
New recording
Process
Session
Program
Test data
Function module
Database selection for recordings

Recording:	ZRECORDING1	frm		to :		Created by	*

Recording	CreatedBy	Date	Time	Transact.	Screens
ZRECORDING1	DEVELOP	15.01.2002	14:59:27	1	5

DEV (1) (555) SAP04 INS 15:07

Recording Step 5: Enter the program name .


Transaction Recorder: Recording Overview
Recording Edit Goto System Help
New recording
Process
Session
Program
Test data
Function module

Database selection for recordings

Recording: ZRECORDING1 frm to : Created by *

Recording	CreatedBy	Date	Time	Transact.	Screens
ZRECO					

Generate Program for Recording ZRECORDING1

Program name ZREPORT1

Field contents

(● Read from file) ○ Transfer from recording

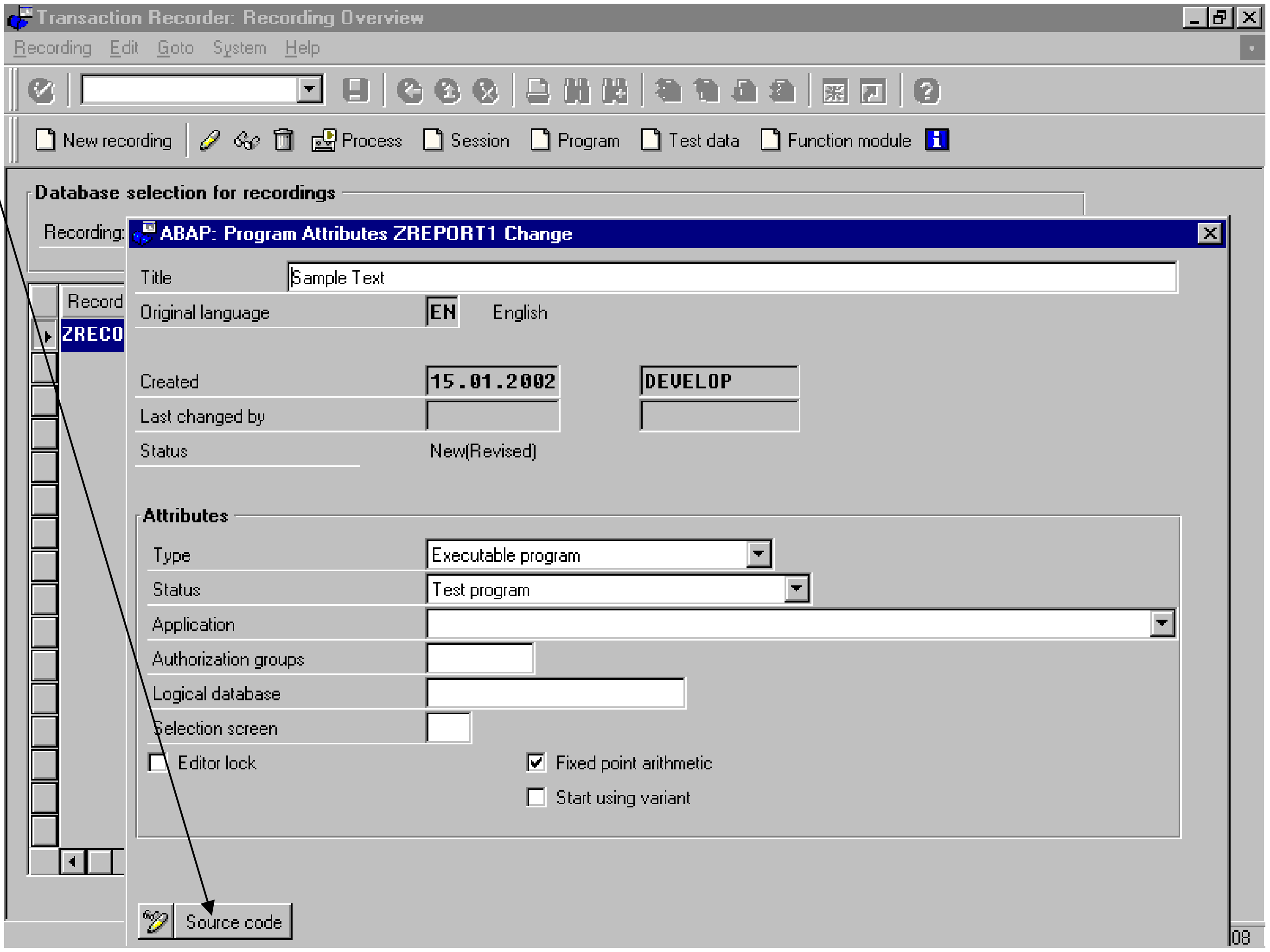
Test data

☐ Create

File name

DEV (1) (555) SAP04 INS 15:07

Recording Step 6: Enter the program attributes. Click "Source Code" Button.


Transaction Recorder: Recording Overview
Recording Edit Goto System Help
New recording
Process
Session
Program
Test data
Function module
Database selection for recordings
Recording:
Record
ZRECO
08
Source code

ABAP: Program Attributes ZREPORT1 Change

Title	Sample Text	
Original language	EN	English
Created	15.01.2002	DEVELOP
Last changed by		
Status	New(Revised)	

Attributes

Type	Executable program
Status	Test program
Application	
Authorization groups	
Logical database	
Selection screen	

☐ Editor lock
☑ Fixed point arithmetic
☐ Start using variant

And the code generated was as below.

```
 PERFORM BDC_DYNPRO      USING 'SAPMF02K' '0107'.
  PERFORM BDC_FIELD       USING 'BDC_CURSOR'      'RF02K-KTOKK'.
  PERFORM BDC_FIELD       USING 'BDC_OKCODE'   '/00'.
  PERFORM BDC_FIELD       USING 'RF02K-LIFNR' RECORD-LIFNR_001.
  PERFORM BDC_FIELD       USING 'RF02K-KTOKK' RECORD-KTOKK_002.
  PERFORM BDC_DYNPRO      USING 'SAPMF02K' '0110'.
  PERFORM BDC_FIELD       USING 'BDC_CURSOR' 'LFA1-SORTL'.
  PERFORM BDC_FIELD       USING 'BDC_OKCODE'  '/00'.
  PERFORM BDC_FIELD       USING 'LFA1-NAME1'  RECORD-NAME1_003.
  PERFORM BDC_FIELD       USING 'LFA1-SORTL' RECORD-SORTL_004.
  PERFORM BDC_FIELD       USING 'LFA1-LAND1'  RECORD-LAND1_005.
  PERFORM BDC_FIELD       USING 'LFA1-SPRAS'  RECORD-SPRAS_006.
  PERFORM BDC_DYNPRO      USING 'SAPMF02K' '0120'.
  PERFORM BDC_FIELD       USING 'BDC_CURSOR  'LFA1-KUNNR'.
  PERFORM BDC_FIELD       USING 'BDC_OKCODE' '/00'.
  PERFORM BDC_DYNPRO      USING 'SAPMF02K' '0130'.
  PERFORM BDC_FIELD       USING 'BDC_CURSOR' 'LFBK-BANKS(01)'.
  PERFORM BDC_FIELD       USING 'BDC_OKCODE' '=ENTR'.
  PERFORM BDC_DYNPRO      USING 'SAPLSPO1' '0300'.
  PERFORM BDC_FIELD       USING 'BDC_OKCODE'   '=YES'.
  PERFORM BDC_TRANSACTION USING 'MK01'.
```

Sample BDC Program (Session Method):

```
REPORT Zreport1.
DATA:   I_BDCDATA LIKE BDCDATA OCCURS 0 WITH HEADER LINE.
DATA: BEGIN OF RECORD OCCURS 0, “Declaration of the data that is to be uploaded from
                                                                         the file
        LIFNR_001(016),
        KTOKK_002(004),
        NAME1_003(035),
        SORTL_004(010),
        LAND1_005(003),
        SPRAS_006(002),
      END OF RECORD.
START-OF-SELECTION.
 PERFORM OPEN_GROUP.
*Uploading data from the local file C:\Vendor1.txt
  CALL FUNCTION 'WS_UPLOAD'
   EXPORTING
     FILENAME     = 'C:\VENDOR1.TXT '
     FILETYPE     = 'DAT'
   TABLES
       DATA_TAB  = RECORD.
  IF SY-SUBRC <> 0.
    WRITE 'ERROR IN UPLOAD'.
  ENDIF.
```

Sample BDC Program (Continues):

```
LOOP AT RECORD. “Filling the BDC table with data

   PERFORM BDC_DYNPRO      USING 'SAPMF02K' '0107'.
   PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                                 'RF02K-KTOKK'.
   PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                                 '/00'.
   PERFORM BDC_FIELD       USING 'RF02K-LIFNR'
                                 RECORD-LIFNR_001.
   PERFORM BDC_FIELD       USING 'RF02K-KTOKK'
                                 RECORD-KTOKK_002.
   PERFORM BDC_DYNPRO      USING 'SAPMF02K' '0110'.
   PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                                 'LFA1-SORTL'.
   PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                                 '/00'.
   PERFORM BDC_FIELD       USING 'LFA1-NAME1'
                                 RECORD-NAME1_003.
   PERFORM BDC_FIELD       USING 'LFA1-SORTL'
                                 RECORD-SORTL_004.
   PERFORM BDC_FIELD       USING 'LFA1-LAND1'
                                 RECORD-LAND1_005.
```

Sample BDC Program (Continues):

```
  PERFORM BDC_FIELD       USING 'LFA1-SPRAS'
                                RECORD-SPRAS_006.
  PERFORM BDC_DYNPRO      USING 'SAPMF02K' '0120'.
  PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                                'LFA1-KUNNR'.
  PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                                '/00'.
  PERFORM BDC_DYNPRO      USING 'SAPMF02K' '0130'.
  PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                                'LFBK-BANKS(01)'.
  PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                                '=ENTR'.
  PERFORM BDC_DYNPRO      USING 'SAPLSPO1' '0300'.
  PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                                '=YES'.
  PERFORM BDC_TRANSACTION USING 'MK01'.

 ENDLOOP.
PERFORM CLOSE_GROUP. " Closing the BDC session
```

Sample BDC Program (Continues):

```
 FORM OPEN_GROUP.
  CALL FUNCTION 'BDC_OPEN_GROUP'
       EXPORTING
            CLIENT   = SY-MANDT
            GROUP    = 'SESSION1'
            USER     = SY-UNAME
            KEEP     = 'X'.
  IF SY-SUBRC <> 0.
WRITE 'ERROR IN OPEN_GROUP'.
ENDIF.
ENDFORM.

FORM BDC_DYNPRO USING PROGRAM DYNPRO.
  CLEAR I_BDCDATA.
  I_BDCDATA-PROGRAM  = PROGRAM.
  I_BDCDATA-DYNPRO   = DYNPRO.
  I_BDCDATA-DYNBEGIN = 'X'.
  APPEND I_BDCDATA.
ENDFORM.
```

Sample BDC Program (Continues):

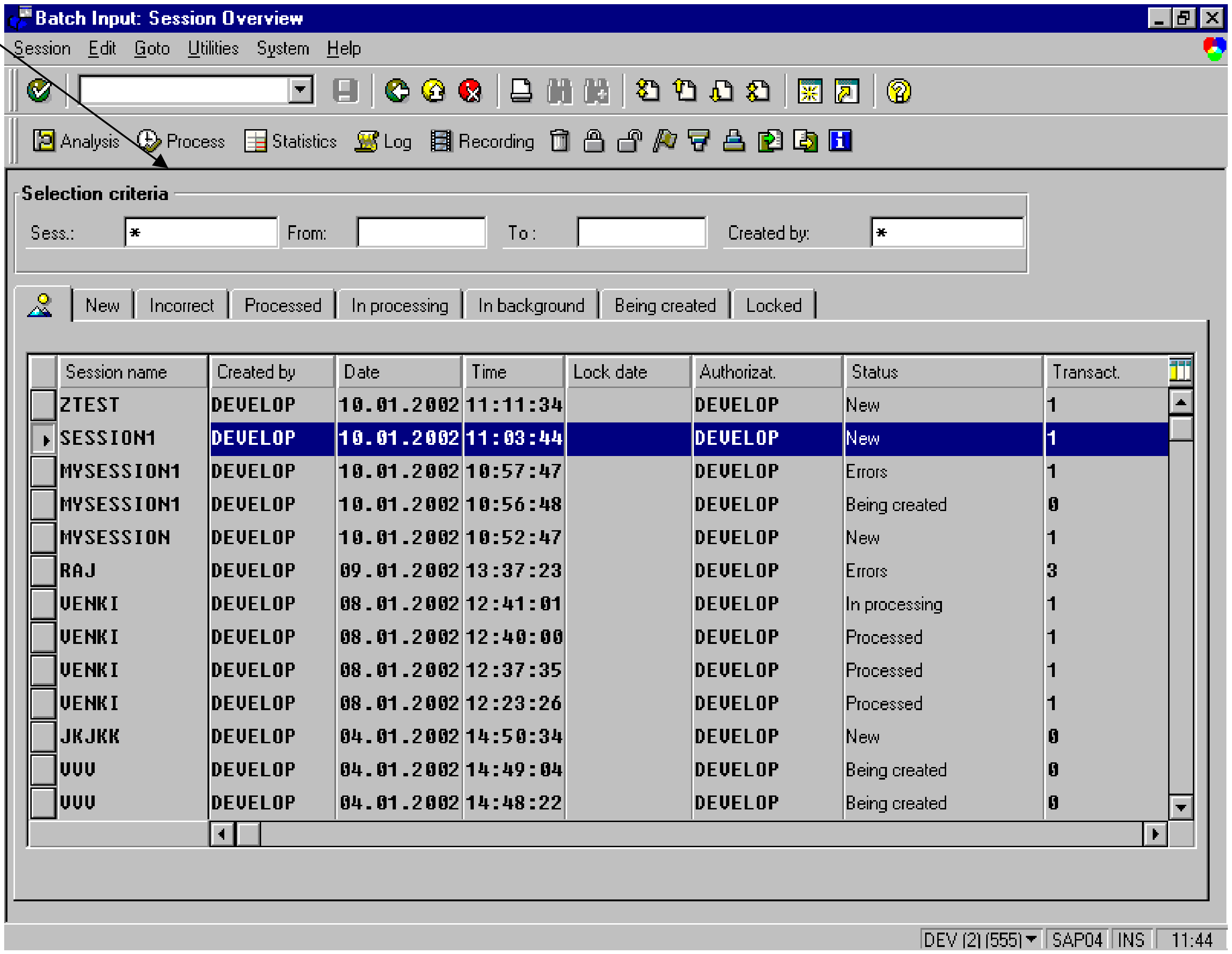
```
FORM BDC_FIELD USING FNAM FVAL.
   CLEAR I_BDCDATA.
   I_BDCDATA-FNAM = FNAM.
   I_BDCDATA-FVAL = FVAL.
   APPEND I_BDCDATA.
ENDFORM.

FORM BDC_TRANSACTION USING TCODE.
CALL FUNCTION 'BDC_INSERT'
       EXPORTING TCODE     = TCODE
       TABLES    DYNPROTAB = I_BDCDATA.
ENDFORM.

FORM CLOSE_GROUP.
CALL FUNCTION 'BDC_CLOSE_GROUP'.
ENDFORM.                    " CLOSE_GROUP
```

Session Overview Screen (Transaction Code : SM35) Example transaction is MK01(Vendor Creation)

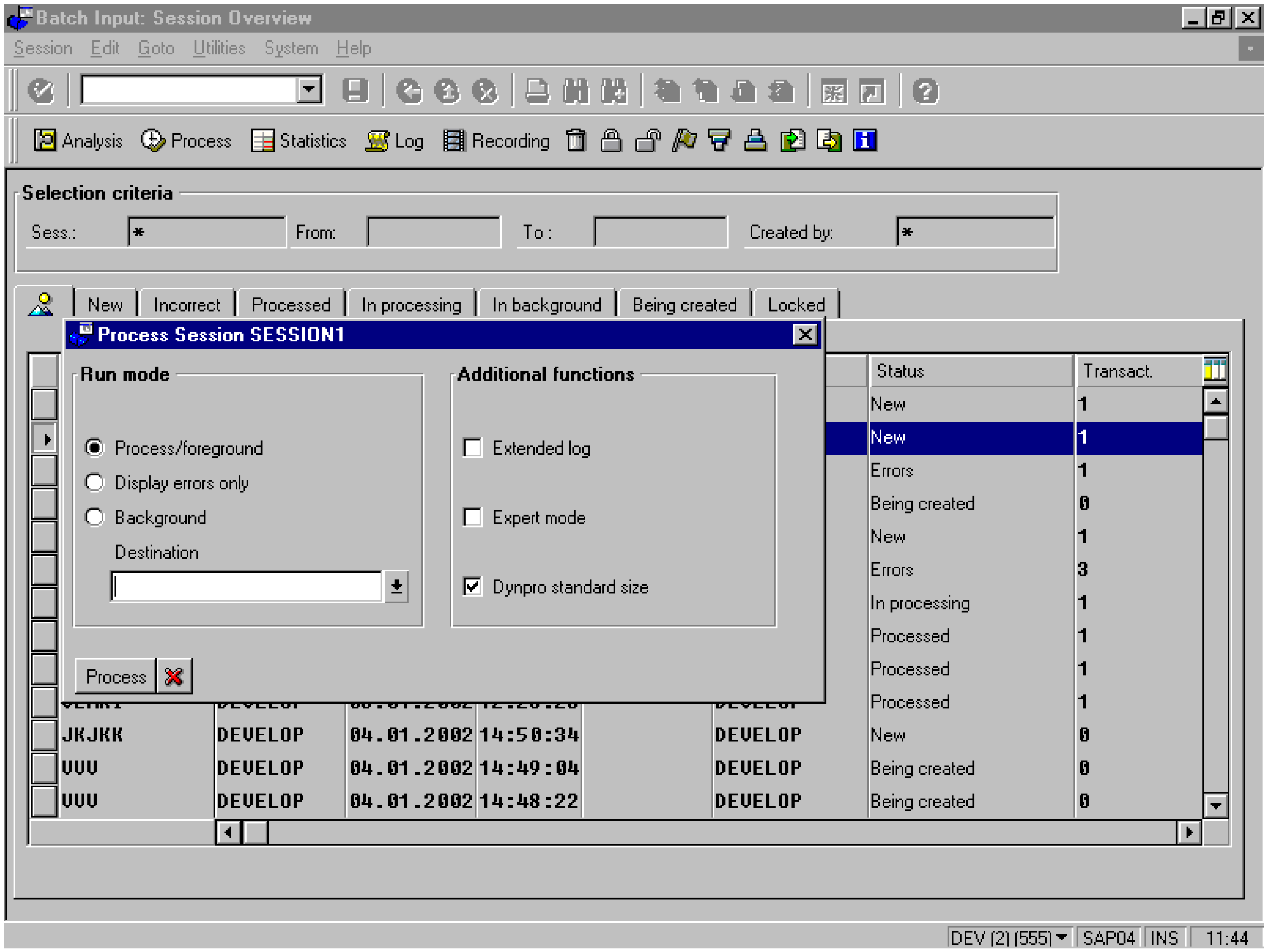
Click "Process" Button


Batch Input: Session Overview
Session Edit Goto Utilities System Help
Analysis Process Statistics Log Recording
Selection criteria
New Incorrect Processed In processing In background Being created Locked
DEV (2) (555) SAP04 INS 11:44

Sess.: * From: To : Created by: *

Session name	Created by	Date	Time	Lock date	Authorizat.	Status	Transact.
ZTEST	DEVELOP	10.01.2002	11:11:34		DEVELOP	New	1
SESSION1	DEVELOP	10.01.2002	11:03:44		DEVELOP	New	1
MYSESSION1	DEVELOP	10.01.2002	10:57:47		DEVELOP	Errors	1
MYSESSION1	DEVELOP	10.01.2002	10:56:48		DEVELOP	Being created	0
MYSESSION	DEVELOP	10.01.2002	10:52:47		DEVELOP	New	1
RAJ	DEVELOP	09.01.2002	13:37:23		DEVELOP	Errors	3
VENKI	DEVELOP	08.01.2002	12:41:01		DEVELOP	In processing	1
VENKI	DEVELOP	08.01.2002	12:40:00		DEVELOP	Processed	1
VENKI	DEVELOP	08.01.2002	12:37:35		DEVELOP	Processed	1
VENKI	DEVELOP	08.01.2002	12:23:26		DEVELOP	Processed	1
JKJKK	DEVELOP	04.01.2002	14:50:34		DEVELOP	New	0
VVV	DEVELOP	04.01.2002	14:49:04		DEVELOP	Being created	0
VVV	DEVELOP	04.01.2002	14:48:22		DEVELOP	Being created	0

Enter the processing options (like Foreground/ Background etc..)


Batch Input: Session Overview
Session Edit Goto Utilities System Help
Analysis Process Statistics Log Recording
New Incorrect Processed In processing In background Being created Locked

Selection criteria

Sess.: * From: To: Created by: *

Process Session SESSION1

Run mode

- [✓ Process/foreground]
- Display errors only
- Background

Destination

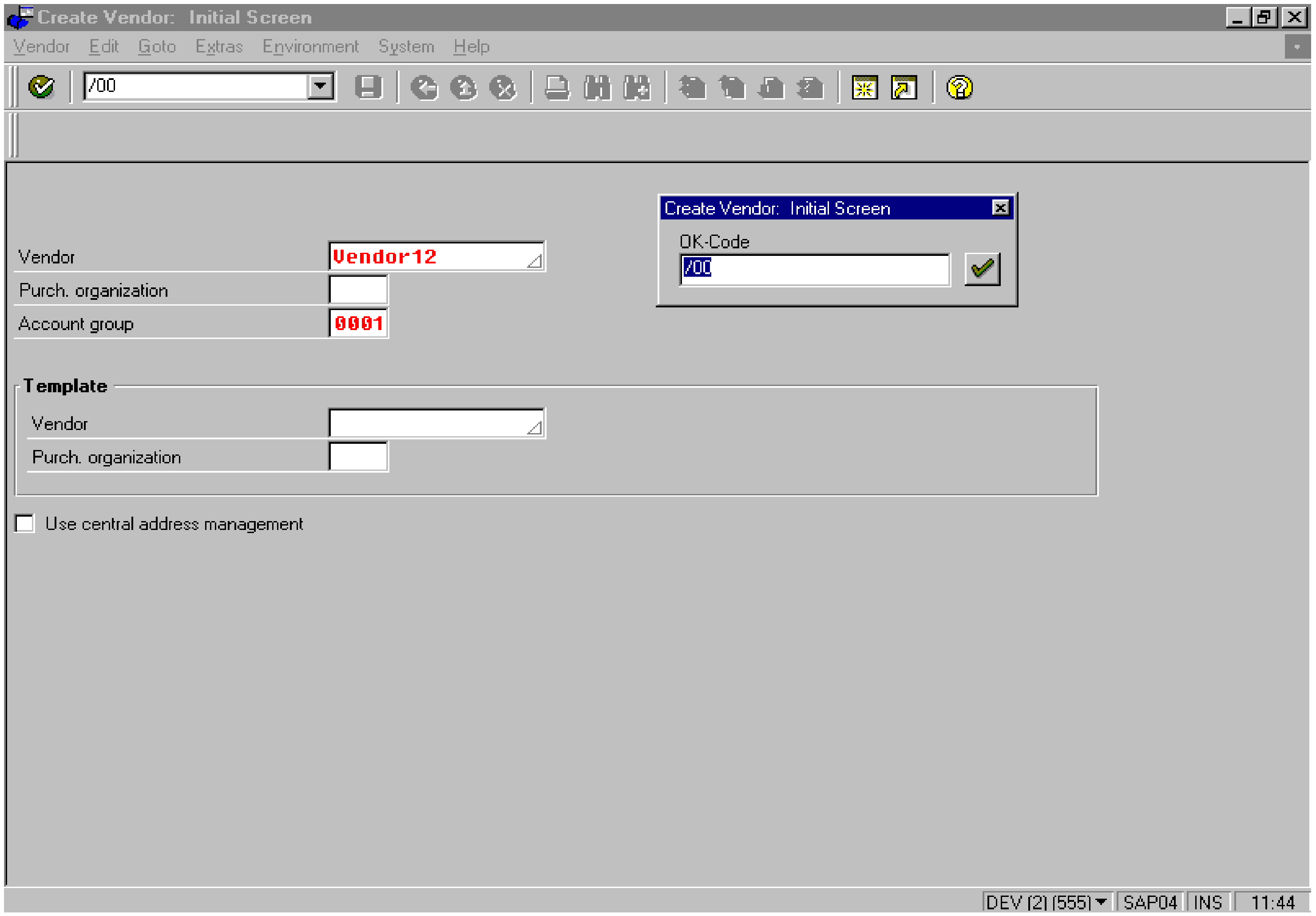
Additional functions

- [ ] Extended log
- [ ] Expert mode
- [✓] Dynpro standard size

						Status	Transact.
						New	1
						New	1
						Errors	1
						Being created	0
						New	1
						Errors	3
						In processing	1
						Processed	1
						Processed	1
[illegible]	[illegible]	[illegible]	[illegible]		[illegible]	Processed	1
JKJKK	DEVELOP	04.01.2002	14:50:34		DEVELOP	New	0
VVV	DEVELOP	04.01.2002	14:49:04		DEVELOP	Being created	0
VVV	DEVELOP	04.01.2002	14:48:22		DEVELOP	Being created	0

Foreground Processing:

First screen of the transaction MK01(Vendor Creation)


Create Vendor: Initial Screen
Vendor Edit Goto Extras Environment System Help

Create Vendor: Initial Screen

OK-Code: /00

Vendor	Vendor12
Purch. organization	
Account group	0001

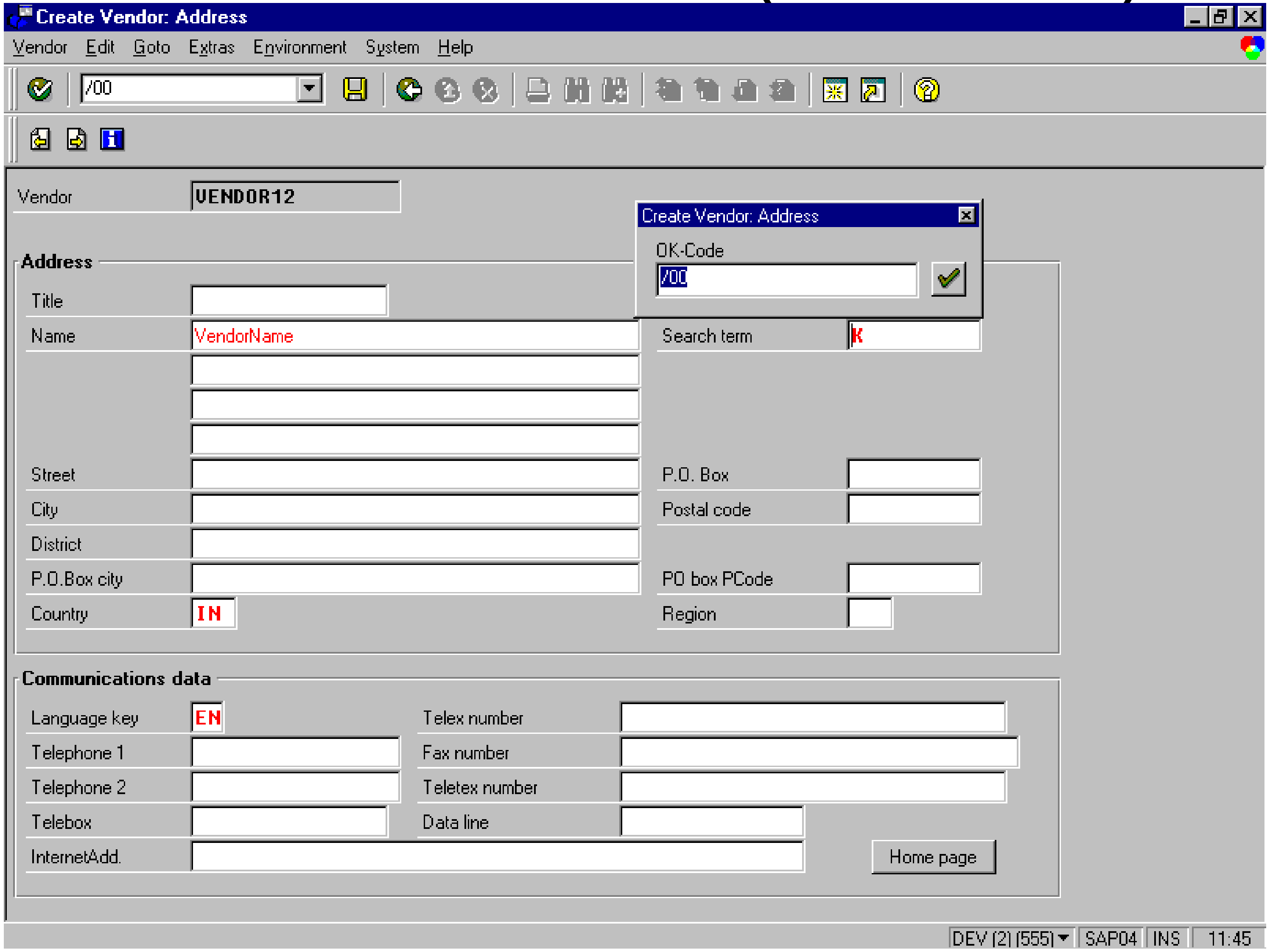
Template

Vendor	
Purch. organization	

☐ Use central address management

DEV (2) (555) | SAP04 | INS | 11:44

Second screen of the Transaction MK01 (Vendor Creation)



Vendor VENDOR12

OK-Code /00

Address

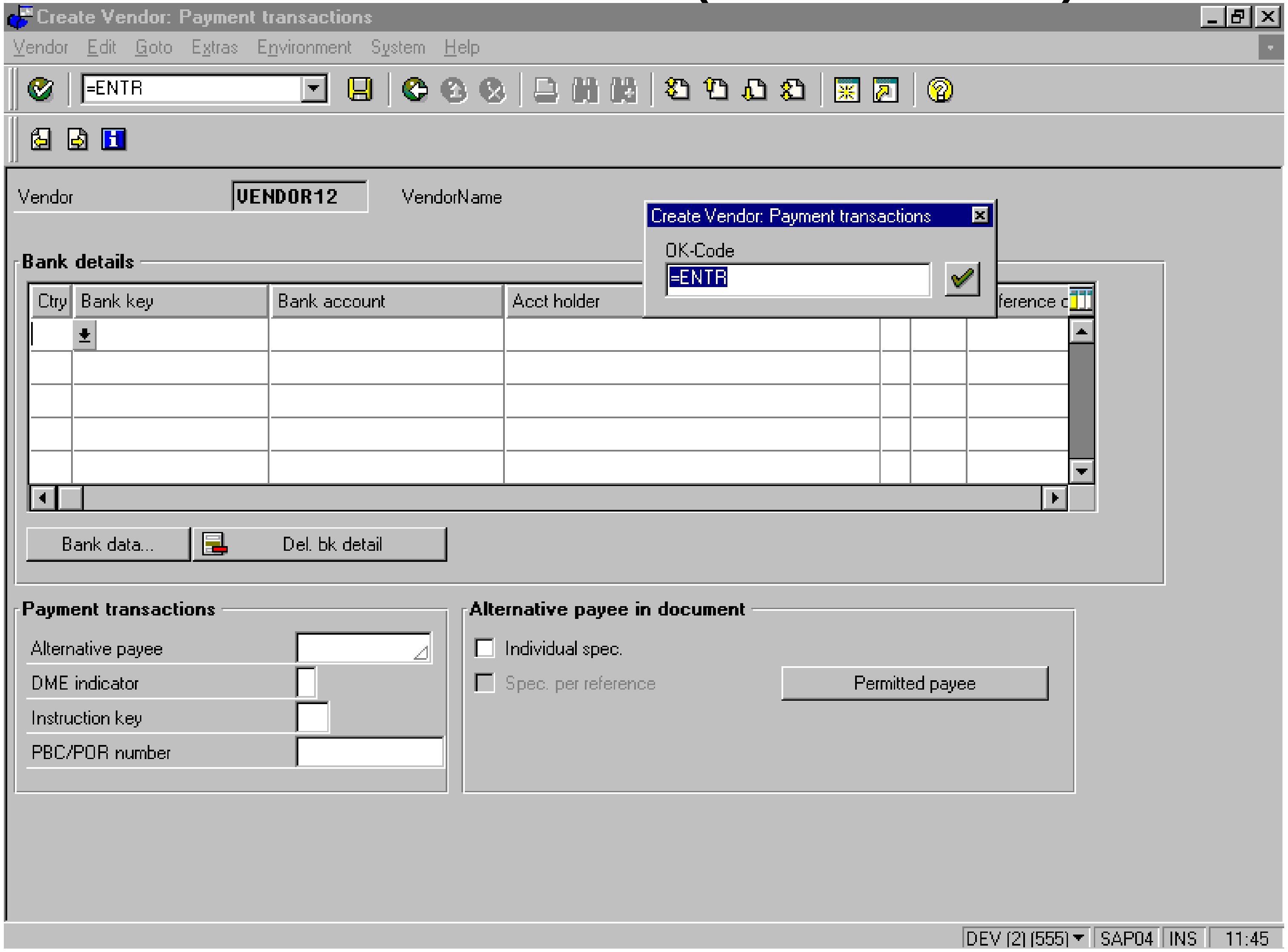
Title			
Name	VendorName	Search term	K
Street		P.O. Box	
City		Postal code	
District			
P.O.Box city		PO box PCode	
Country	IN	Region	

Communications data

Language key	EN	Telex number	
Telephone 1		Fax number	
Telephone 2		Teletex number	
Telebox		Data line	
InternetAdd.			

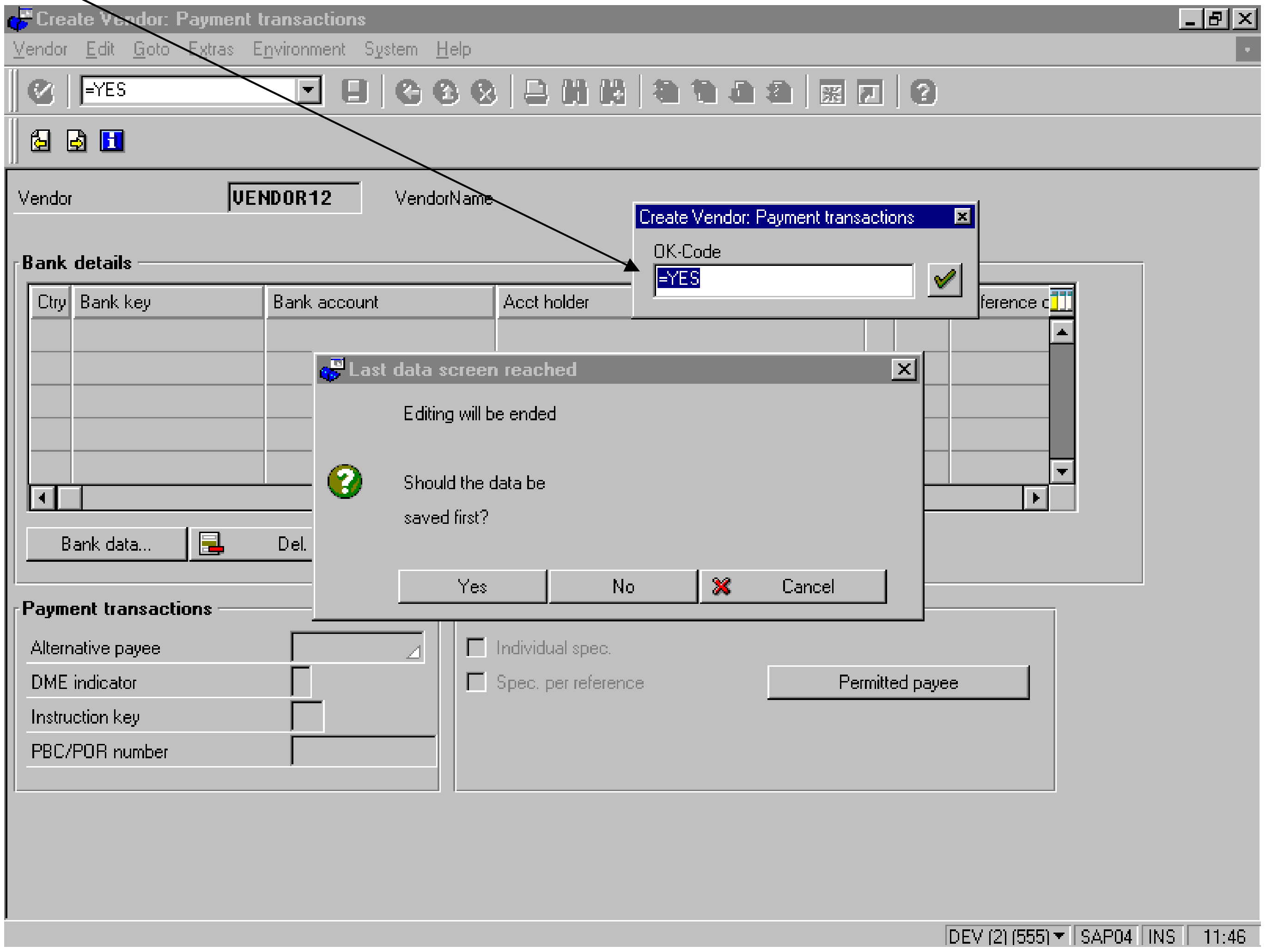
DEV (2) (555) | SAP04 | INS | 11:45

Last Screen of the Transaction MK01 (Vendor Creation):



Prompt for Saving the record:

Here OK_CODE = YES


Create Vendor: Payment transactions

Vendor Edit Goto Extras Environment System Help

=YES

Vendor VENDOR12 VendorName

Bank details

Ctry | Bank key | Bank account | Acct holder | ference c

Bank data... Del.

Payment transactions

Alternative payee

DME indicator

Instruction key

PBC/POR number

Individual spec.

Spec. per reference

Permitted payee

Create Vendor: Payment transactions

OK-Code

=YES

Last data screen reached

Editing will be ended

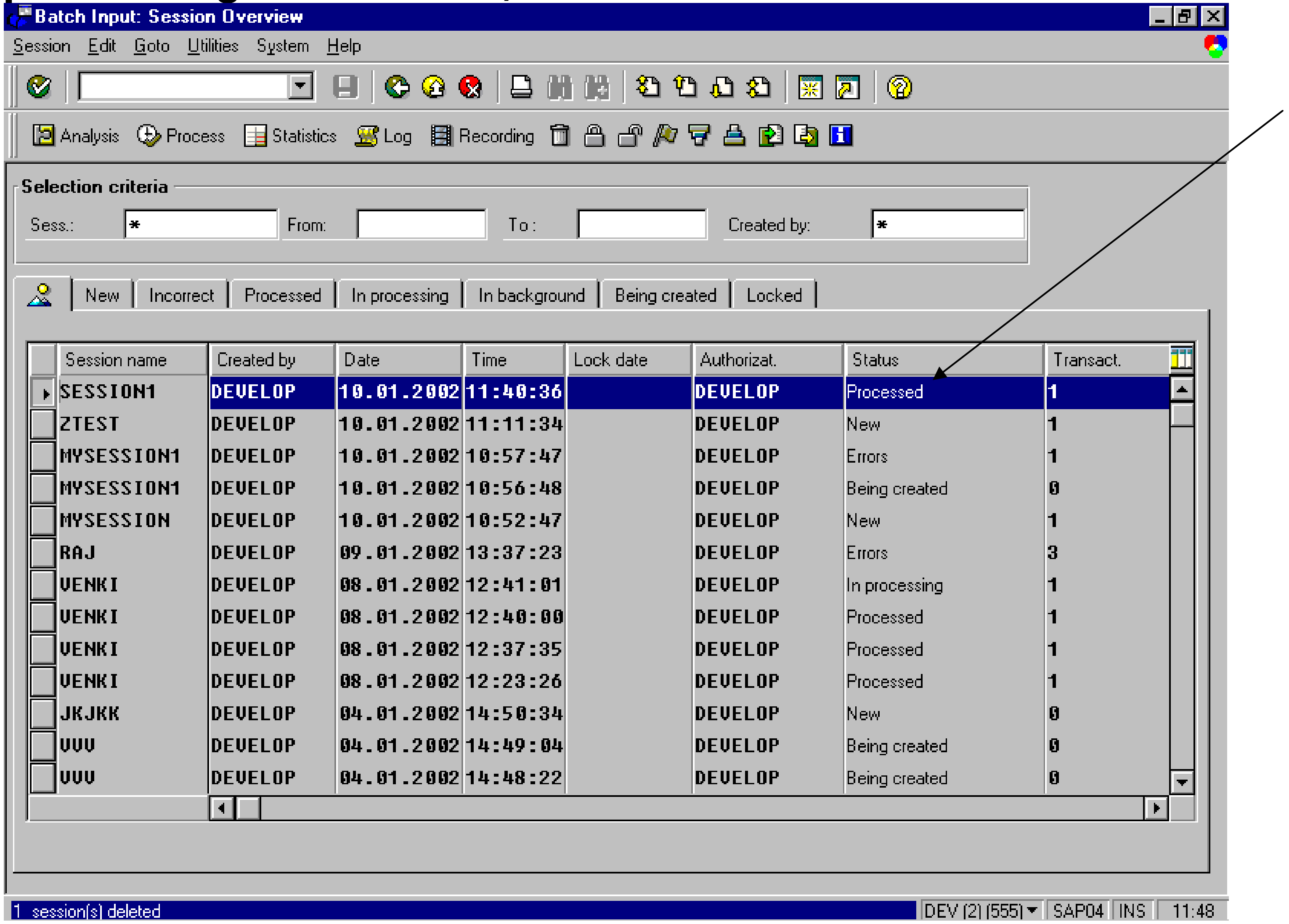
Should the data be

saved first?

Yes No Cancel

DEV (2) (555) SAP04 INS 11:46

After processing the session, the status of the session is “Processed”


Batch Input: Session Overview
Session Edit Goto Utilities System Help
Analysis Process Statistics Log Recording
Selection criteria
New Incorrect Processed In processing In background Being created Locked

Sess.:	*	From:		To :		Created by:	*

Session name	Created by	Date	Time	Lock date	Authorizat.	Status	Transact.
SESSION1	DEVELOP	10.01.2002	11:40:36		DEVELOP	Processed	1
ZTEST	DEVELOP	10.01.2002	11:11:34		DEVELOP	New	1
MYSESSION1	DEVELOP	10.01.2002	10:57:47		DEVELOP	Errors	1
MYSESSION1	DEVELOP	10.01.2002	10:56:48		DEVELOP	Being created	0
MYSESSION	DEVELOP	10.01.2002	10:52:47		DEVELOP	New	1
RAJ	DEVELOP	09.01.2002	13:37:23		DEVELOP	Errors	3
VENKI	DEVELOP	08.01.2002	12:41:01		DEVELOP	In processing	1
VENKI	DEVELOP	08.01.2002	12:40:00		DEVELOP	Processed	1
VENKI	DEVELOP	08.01.2002	12:37:35		DEVELOP	Processed	1
VENKI	DEVELOP	08.01.2002	12:23:26		DEVELOP	Processed	1
JKJKK	DEVELOP	04.01.2002	14:50:34		DEVELOP	New	0
VVV	DEVELOP	04.01.2002	14:49:04		DEVELOP	Being created	0
VVV	DEVELOP	04.01.2002	14:48:22		DEVELOP	Being created	0

1 session(s) deleted | DEV (2) (555) | SAP04 | INS | 11:48

CALL TRANSACTION METHOD :

This is another method to transfer data from the legacy system.

Features:

- Synchronous processing. The system performs a database commit immediately before and after the CALL TRANSACTION USING statement.
- Updating the database can be either synchronous or asynchronous. The program specifies the update type.
- Transfer data for a single transaction.
- Transfers data for a sequence of dialog screens.
- No batch input processing log is generated.

The CALL TRANSACTION Statement

```
CALL TRANSACTION        <transaction code>

     USING        <BDC table>
     MODE         <display mode>
     UPDATE       <update mode>
     MESSAGES     INTO <messtab>
```

<display mode>:

A	Display all
E	Display only if there are errors
N	Display nothing

<update mode>:

S	Do not continue processing until update has finished (synchronous)
A	Continue processing immediately

Sample Program (CALL TRANSACTION):

```
REPORT ZREPORT1.
DATA:   I_BDCDATA LIKE BDCDATA OCCURS 0 WITH HEADER LINE.
DATA: BEGIN OF RECORD OCCURS 0,
        LIFNR_001(016),
        KTOKK_002(004),
        NAME1_003(035),
        SORTL_004(010),
        LAND1_005(003),
        SPRAS_006(002),
      END OF RECORD.
START-OF-SELECTION.
*Uploading data from the local file C:\Vendor1.txt
  CALL FUNCTION 'WS_UPLOAD'
   EXPORTING
     FILENAME     = 'C:\VENDOR11.TXT '
     FILETYPE     = 'DAT'
   TABLES
      DATA_TAB  = RECORD.
  IF SY-SUBRC <> 0.
    WRITE 'ERROR IN UPLOAD'.
  ENDIF.
```

Sample Program (CALL TRANSACTION):

```
 LOOP AT RECORD.
   PERFORM BDC_DYNPRO      USING 'SAPMF02K' '0107'.
   PERFORM BDC_FIELD       USING 'BDC_CURSOR'      'RF02K-KTOKK'.
   PERFORM BDC_FIELD       USING 'BDC_OKCODE’   '/00'.
   PERFORM BDC_FIELD       USING 'RF02K-LIFNR’ RECORD-LIFNR_001.
   PERFORM BDC_FIELD       USING 'RF02K-KTOKK’ RECORD-KTOKK_002.
   PERFORM BDC_DYNPRO      USING 'SAPMF02K' '0110'.
   PERFORM BDC_FIELD       USING 'BDC_CURSOR’ 'LFA1-SORTL'.
   PERFORM BDC_FIELD       USING 'BDC_OKCODE’  '/00'.
   PERFORM BDC_FIELD       USING 'LFA1-NAME1’  RECORD-NAME1_003.
   PERFORM BDC_FIELD       USING 'LFA1-SORTL’ RECORD-SORTL_004.
   PERFORM BDC_FIELD       USING 'LFA1-LAND1'  RECORD-LAND1_005.
   PERFORM BDC_FIELD       USING 'LFA1-SPRAS'  RECORD-SPRAS_006.
   PERFORM BDC_DYNPRO      USING 'SAPMF02K' '0120'.
   PERFORM BDC_FIELD       USING 'BDC_CURSOR  'LFA1-KUNNR'.
   PERFORM BDC_FIELD       USING 'BDC_OKCODE’ '/00'.
   PERFORM BDC_DYNPRO      USING 'SAPMF02K' '0130'.
   PERFORM BDC_FIELD       USING 'BDC_CURSOR’ 'LFBK-BANKS(01)'.
   PERFORM BDC_FIELD       USING 'BDC_OKCODE’ '=ENTR'.
   PERFORM BDC_DYNPRO      USING 'SAPLSPO1' '0300'.
   PERFORM BDC_FIELD       USING 'BDC_OKCODE’   '=YES'.
   PERFORM BDC_TRANSACTION USING 'MK01'.
  ENDLOOP.
```

Sample Program (CALL TRANSACTION):

```
FORM BDC_DYNPRO USING PROGRAM DYNPRO.
  CLEAR I_BDCDATA.
  I_BDCDATA-PROGRAM  = PROGRAM.
  I_BDCDATA-DYNPRO   = DYNPRO.
  I_BDCDATA-DYNBEGIN = 'X'. APPEND I_BDCDATA.
ENDFORM.

FORM BDC_FIELD USING FNAM FVAL.
    CLEAR I_BDCDATA.
    I_BDCDATA-FNAM = FNAM.
    I_BDCDATA-FVAL = FVAL. APPEND I_BDCDATA.
ENDFORM.

FORM BDC_TRANSACTION USING TCODE.
CALL TRANSACTION TCODE USING I_BDCDATA
                     MODE   'A'. "Processing in fore-ground
ENDFORM.
```

Batch Input / CALL TRANSACTION - Comparision

	Session	CALL TRANSACTION
Return code	No	Yes
Database Update	Synchronous	Asynchronous/Synchronous
Processing	Time-delayed	Immediately
Transactions	More than one	Only One
Error Log	Will be created	Will not be created

Exercise:

Write a batch input program for transaction MM01 using following data from a local file.

Material type	Material Group	Basic Data1	Basic Data2	Description	Unit Of Measure
M	BOH	X	X	Material1	KG

Solution:

```
REPORT ZREPORT1.
DATA:   I_BDCDATA LIKE BDCDATA OCCURS 0 WITH HEADER LINE.
DATA: BEGIN OF RECORD OCCURS 0,
MBRSH_001(001),
MTART_002(004),
KZSEL_01_003(001),
KZSEL_02_004(001),
MAKTX_005(040),
MEINS_006(003),
 MAKTX_007(040),
 END OF RECORD.
START-OF-SELECTION.
*Uploading data from the local file C:\Vendor1.txt
  CALL FUNCTION 'WS_UPLOAD'
   EXPORTING
     FILENAME     = 'C:\MARA.TXT '
     FILETYPE     = 'DAT'
   TABLES
      DATA_TAB  = RECORD.
```

```
IF SY-SUBRC <> 0.
   WRITE 'ERROR IN UPLOAD'.
 ENDIF.
PERFORM OPEN_GROUP.
LOOP AT RECORD.
   PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '0060'.
   PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                                 'RMMG1-MTART'.
   PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                                 '/00'.
   PERFORM BDC_FIELD       USING 'RMMG1-MBRSH'
                                 RECORD-MBRSH_001.
   PERFORM BDC_FIELD       USING 'RMMG1-MTART'
                                 RECORD-MTART_002.
   PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '0070'.
   PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                                 'MSICHTAUSW-DYTXT(02)'.
   PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                                 '=ENTR'.
```

```
PERFORM BDC_FIELD       USING 'MSICHTAUSW-KZSEL(01)'
                              RECORD-KZSEL_01_003.
   PERFORM BDC_FIELD       USING 'MSICHTAUSW-KZSEL(02)'
                              RECORD-KZSEL_02_004.
   PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '4004'.
   PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                              '/00'.
   PERFORM BDC_FIELD       USING 'MAKT-MAKTX'
                              RECORD-MAKTX_005.
   PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                              'MARA-MEINS'.
   PERFORM BDC_FIELD       USING 'MARA-MEINS'
                              RECORD-MEINS_006.
   PERFORM BDC_DYNPRO      USING 'SAPLMGMM' '4004'.
   PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                              '/00'.
   PERFORM BDC_FIELD       USING 'BDC_CURSOR'
                              'MAKT-MAKTX'.
```

```
    PERFORM BDC_FIELD       USING 'MAKT-MAKTX'
                                  RECORD-MAKTX_007.
    PERFORM BDC_DYNPRO      USING 'SAPLSPO1' '0300'.
    PERFORM BDC_FIELD       USING 'BDC_OKCODE'
                                  '=YES'.
    PERFORM BDC_TRANSACTION USING 'MM01'.
ENDLOOP.
 PERFORM CLOSE_GROUP.
FORM OPEN_GROUP.
  CALL FUNCTION 'BDC_OPEN_GROUP'
       EXPORTING
            CLIENT   = SY-MANDT
            GROUP    = 'SESSION1'
            USER     = SY-UNAME
            KEEP     = 'X'.
  IF SY-SUBRC <> 0.
WRITE 'ERROR IN OPEN_GROUP'.
ENDIF.
ENDFORM.
```

```
FORM BDC_DYNPRO USING PROGRAM DYNPRO.
  CLEAR I_BDCDATA.
  I_BDCDATA-PROGRAM  = PROGRAM.
  I_BDCDATA-DYNPRO   = DYNPRO.
  I_BDCDATA-DYNBEGIN = 'X'.
  APPEND I_BDCDATA.
ENDFORM.
FORM BDC_FIELD USING FNAM FVAL.
    CLEAR I_BDCDATA.
    I_BDCDATA-FNAM = FNAM.
    I_BDCDATA-FVAL = FVAL.
    APPEND I_BDCDATA.
ENDFORM.

FORM BDC_TRANSACTION USING TCODE.
CALL FUNCTION 'BDC_INSERT'
        EXPORTING TCODE     = TCODE
        TABLES    DYNPROTAB = I_BDCDATA.
ENDFORM.
FORM CLOSE_GROUP.
CALL FUNCTION 'BDC_CLOSE_GROUP'.
ENDFORM.                    " CLOSE_GROUP
```

ABAP/4 allows us to work with sequential files

- **on the Application server**
- **on the Presentation server**

WORKING WITH FILES ON THE APPLICATION SERVER:

ABAP/4 provides three statements for handling files:

- **OPEN DATASET**
- **CLOSE DATASET**
- **DELETE DATASET**
- **READ DATASET**
- **TRANSFER**

OPEN DATASET

Opens the specified file. If you do not use any additions, the file is opened for reading in binary mode. It returns SY-SUBRC = 0 if the file is opened successfully. Otherwise SY-SUBRC = 8.

Syntax

OPEN DATASET <dsn> [Additions].

Additions:

1. FOR INPUT (Default)
2. FOR OUTPUT
3. FOR APPENDING
4. IN BINARY MODE
5. IN TEXT MODE
6. AT POSITION p
7. TYPE ctrl
8. MESSAGE mess
9. FILTER f

1. OPEN DATASET <dsn> FOR INPUT.

   This statement tries to open the field in 'read/update' mode (as long as the user has write authorization).If the user does not have write authorization, the system opens the file in 'read' mode. If this fails, an error occurs.

2. OPEN DATASET <dsn> FOR OUTPUT.

   This statement tries to open the file in 'write/update' mode as long as the user has read authorization. If the authorization is missing, the system opens the file in 'write' mode. If the file already exists, its existing content is deleted. If the file does not exist, the system creates it.

3. OPEN DATASET <dsn> FOR APPENDING.

This statement tries to open the file in 'append' mode. If the file is already open, the system moves to the end of the file. When you open a file using FOR APPENDING, attempting to read the file sets SY-SUBRC to 4. The system display the end of the file.

Note :

You can only use one of the additions 1 to 3 in a single statement

4. OPEN DATASET <dsn> IN BINARY MODE.

The contents of the file are not structured in lines in the READ DATASET or TRANSFER operations. Instead, they are input or output as a stream. You do not have to specify the IN BINARY MODE addition explicitly.

5. OPEN DATASET <dsn> IN TEXT MODE.

If you use this addition, the contents of the file are structured in lines. Each time you use the READ DATASET or TRANSFER statement, the system reads or writes a single line. If the data object to which you are transferring the data is too big, it is padded with spaces. If it is too small, the data record is truncated.

Note

You can only use one of additions 4 and 5 in a single statement.

6. OPEN DATASET <dsn> AT POSITION p.

Use this addition to specify the explicit starting position p in the file (calculated in bytes from the start of the file). The next read or write operation will start at this position. You cannot position before the beginning of the file. Do not use this addition with the IN TEXT MODE addition, since the physical representation of a text file depends heavily on the underlying operating system.

If you use OPEN ... FOR OUTPUT AT POSITION ..., the contents of the file are destroyed if the file already existed. To avoid this, use OPEN ... FOR INPUT AT POSITION ... instead.

Note

OPEN ... AT POSITION p does not work for file positions where p >= 2 giga bytes.

7. OPEN DATASET <dsn> TYPE ctrl .

You can use the ctrl field to specify further file attributes. The contents of this field are passed unchanged and unchecked to the operating system. The syntax for the attributes is dependent on the operating system.

8. OPEN DATASET <dsn> MESSAGE msg.

If an error occurs while the file is being opened, the corresponding operating system message is placed in field msg.

Example

```
DATA: dsn(20) VALUE '/usr/test.dat',
                              msg(100).
OPEN DATASET dsn FOR INPUT MESSAGE msg.
IF sy-subrc <> 0.
WRITE / msg.
ENDIF.
```

9. OPEN DATASET <dsn> `FILTER f.`

If you are working under UNIX or Windows NT, you can specify an operating system command in the field f.

Example

Under UNIX, the following statements opens the file dsn and writes the data to the file in compressed form because of the UNIX command 'compress' :

```
DATA dsn(20) VALUE '/usr/test.dat'.
OPEN DATASET dsn FOR OUTPUT FILTER 'compress'.
```

CLOSE DATASET

Closes the specified file.

Syntax

```
CLOSE DATASET <dsn>.
```

DELETE DATASET

Deletes the file specified file. If it deletes the file successfully it returns SY-SUBRC = 0. Otherwise returns SY-SUBRC = 4. The possible reasons for failing are:

- The file does not exist.
- The file is a directory.
- The file is a program that is currently running.

READ DATASET

Used to read a record from a file.

Syntax

```
READ DATASET dsn INTO f.
```

Addition : LENGTH len.

The actual length of the data objet read is placed in the field len after the read access. len must be defined as a variable. A syntax error will occur if you define it as a constant. The following example displays 9.

Example

```
DATA: len TYPE i,
      text(30) TYPE c VALUE 'Beethoven',
      dir(30) TYPE c VALUE '/usr/test.dat'.
      OPEN DATASET dir IN TEXT MODE.
      TRANSFER text TO dir.   CLOSE DATASET dir.
      OPEN DATASET dir IN TEXT MODE.
      READ DATASET dir INTO text LENGTH len.
      CLOSE DATASET dir.  WRITE /  len.
```

TRANSFER statement

Used to write a record into a file.

Syntax

TRANSFER f TO dsn.

Transfers the data object f to a sequential file whose name is specified in dsn. dsn can be a field or a literal. You must already have opened the file. . If the specified file is not already open, TRANSFER attempts to open the file FOR OUTPUT IN BINARY MODE. If this is not possible, a runtime error occurs.f can be a field, a string, or a structure.

Addition : LENGTH len.

The length of the data object to be written is defined by len, where len can be either a constant or a variable. If len is smaller than the length of the data object f, the system truncates character fields (C, N, D, T, X,P, STRING) on the right. With type I or F fields, unexpected results may occur if len is shorter than the default length for the field type.

WORKING WITH FILES ON THE PRESENTATION SERVER:

To work with files on the presentation server , SAP provides some special function modules WS_UPLOAD, for reading from a file, and WS_DOWNLOAD, for writing into the file. An internal table must be used as an interface between the program and the function module.

Writing data to a file on the presentation server:

To write data from an internal table to a file on the presentation server, use function module WS_DOWNLOAD. The most important parameters that are exported are as follows:

BIN_FILESIZE

File Length for binary files. A length of zero or the length which is larger than the number of bytes in the internal table (width * number of lines) causes an exception.

CODEPAGE

Only for download in DOS

FILENAME

The name of the file that is to be generated on the presentation server(if necessary with predefined path name). If the path doesn’t exist or the file cannot be opened, an exception will be raised.

FILETYPE

The target format of the file. Valid values are:

‘ASC’ : ASCII format, the table is stored with rows.

‘DAT’: ASCII format as in 'ASC', additional column separation with TABs.

‘BIN’ : Binary format (specification of BIN_FILESIZE required)

‘DBF’ : Stored as Dbase file (always with DOS code page).

‘IBM’ : ASCII format as in 'ASC' with IBM code page conversion (DOS)

MODE

Writing mode (‘A’ = Append , empty = Overwrite)

FILELENGTH

The length of the generated file is returned.

TABLE PARAMETER

DATA_TAB

The source internal table whose contents are downloaded into a file.

EXCEPTIONS

The exceptions for the function module WS_DOWNLOAD are

FILE_OPEN_ERROR	**The file cannot be opened.**
FILE_WRITE_ERROR	**The data could not be loaded into the file.**
INVALID_FILE_SIZE	**The parameter BIN_FILESIZE is either zero or greater that the table size.**

Reading data from a file on the presentation server:

To read data from the presentation server into an internal table we use the function module WS_UPLOAD. The most important parameters that are exported are as follows:

CODEPAGE

Only for download in DOS.

FILENAME

Name of the file

FILETYPE

The source file type. Valid values are:

‘BIN’ : Binary files.

‘ASC’: ASCII files, text files with end-of-line markers.

‘DAT’: The file is loaded line by line into the transferred table. Tabs in the file mean a change of field.

Export Parameters for WS_UPLOAD:

FILELENGTH - Number of bytes transferred.

Table parameters for WS_UPLOAD:

DATA_TAB - Internal target table, to which the data is loaded.

Exceptions for WS_UPLOAD:

CONVERSION_ERROR	**- Errors in the data conversion.**
FILE_OPEN_ERROR	**- System cannot open file.**
FILE-READ_ERROR	**- System cannot read from file**
INVALID_TABLE_WIDTH	**- Invalid table structure**
INVALID_TYPE	**- Invalid value for parameter FILETYPE**

Summary

- Batch Data Communication (BDC) is the process of transferring data from one SAP System to another SAP system or from a non-SAP system to SAP System.
- Two methods of BDCs are there. Session method and CALL TRANSACTION method.
- Working with files on application server and presentation server.
- ABAP/4 statements OPEN DATASET, READ DATASET, DELETE DATASET, CLOSE DATASET, TRANSFER.
- And special function modules for reading and writing data files on presentation server, WS_UPLOAD and WS_DOWNLOAD.