

■ TABC43 Data Transfer

TABC43

R/3 System
Release 46B
17.06.2000

TABC43 Data Transfer	0-1
Copyright.....	0-2
Section Overview	0-4
Section: Data Transfer.....	1-1
Content: Data Transfer	1-2
Overview of Data Transfer: Contents.....	2-1
Course Overview Diagram	2-2
Overview of Data Transfer	2-3
Data Transfer into SAP System	2-4
Direct Data Transfer.....	2-5
Data Transfer Using Interfaces	2-6
Data Transfer Workbench.....	2-7
Overview of Data Transfer	2-8
Analyzing SAP Transactions.....	2-9
Assigning Data.....	2-10
Formatting Data	2-11
Concept Definition	2-12
Data Transfer Using the DX-WB	2-13
DX-WB: Transfer Procedure	2-14
Data Transfer Using LSMW.....	2-15
LSMW Transfer Procedure	2-16
Data Transfer Overview: Unit Summary	2-17
Overview of Data Transfer Exercises	2-18
Project Concept: Contents	3-1
Course Overview Diagram	3-2
Data Transfer Project	3-3
Data Transfer in R/3.....	3-4
Transfer Methods.....	3-5
Batch Input	3-6
Call Transaction.....	3-7
Direct Input Technique.....	3-8
BAPIs	3-9
Transaction Recorder.....	3-10
Project Concept.....	3-11
Data Transfer Project	3-12
Project Prerequisites.....	3-13
Project Tasks	3-14
Test Procedure.....	3-15
Data Transfer Project	3-16
Business Object Repository - BOR.....	3-17

Project Concept.....	3-19
DX-WB Functions.....	3-20
Mapping Project to the DX-WB	3-21
Tasks and Programs	3-22
Programs and Task Types.....	3-23
Registration in the DX-WB	3-24
Organization of Data Transfer Projects	3-25
Creating Projects.....	3-26
Creating Subproject.....	3-27
Create Run Definition.....	3-28
Create Task	3-29
Define Task Attributes.....	3-30
Starting and Analyzing Run.....	3-31
Log	3-32
Processing Run Problems	3-33
Project Concept : Unit Summary	3-34
Project Concept Exercises.....	3-35
Project Concept Solutions.....	3-38
Principles of Standard Data Transfer.....	4-1
Course Overview Diagram	4-2
Standard Transfer Overview.....	4-3
Interfaces for Standard Data Transfer.....	4-4
Tasks in Standard Transfer	4-5
Documentation from DX-WB	4-6
Converting Data	4-7
Formatting.....	4-8
Mapping	4-9
General Record Layout Structure	4-10
Example of Record Layout Type 0	4-11
General Record Layout Structure	4-12
Example: Record Layout Customer	4-13
Supported Methods	4-14
Principles of Standard Data Transfer: Unit Summary	4-15
DX-WB Tools: Contents.....	5-1
Course Overview Diagram	5-2
DX-WB: Tools	5-3
Function: Create File	5-4
Function: Create File with Data	5-5
Function: Display/Change File	5-6
Function: Copy File.....	5-7
Structure Operations	5-8

Mapping Plan	5-10
NODATA Indicator	5-11
Function: Check File	5-12
Record Layout - Displaying Structure (2).....	5-13
Record Layout - Generating Structure (1).....	5-14
Record Layout - Generating Structure (2).....	5-15
DX-WB Tools: Unit Summary	5-16
DX-WB Tools Exercises.....	5-17
Sequential Files: Contents	6-1
Course Overview Diagram	6-2
Overview of Content - Sequential Files.....	6-3
File Transfer.....	6-4
The File Monitor (AL11)	6-5
Directories in the File Monitor.....	6-6
Files in File Monitor (AL11)	6-7
Overview of Content - Sequential Files (2)	6-8
File Processing.....	6-9
Overview Diagram: Read File	6-10
Overview Diagram: Write File	6-11
Open File	6-12
Open File: Additions.....	6-13
Binary Mode and Text Mode	6-14
Transfer Data Record	6-15
Read Data Record	6-16
Close / Delete File	6-17
The NODATA Character.....	6-18
Initializing Record Layout Structures.....	6-19
Initializing Help Structures.....	6-20
Example of Mapping Program.....	6-21
Example of Formatting.....	6-22
Overview of Content - Sequential Files (3)	6-23
Download and Upload.....	6-24
DOWNLOAD Function Module	6-25
Example: DOWNLOAD.....	6-26
UPLOAD Function Module	6-27
Example: UPLOAD.....	6-28
Overview of Content - Sequential Files (4)	6-29
Customizing Logical File Name	6-30
Logical Path and Logical File	6-31
Reserved Words.....	6-32
FILE_GET_NAME Function Module	6-33
.....	..

Sequential Files Exercises.....	6-35
Sequential Files Solutions.....	6-38
Batch Input Processing.....	7-1
Course Overview Diagram	7-2
Batch Input Processing (1).....	7-3
Overview: Batch Input Processing	7-4
Batch Input Program.....	7-5
Format of Batch Input Session.....	7-6
Creating Batch Input Session	7-7
Structure of BDC Table	7-8
Processing a Batch Input Session	7-9
Batch Input Processing.....	7-10
The Batch Input Monitor.....	7-11
Functions of the Batch Input Monitor.....	7-12
Initial Screen of Batch Input Monitor.....	7-13
Processing Mode.....	7-14
Menu Functions for Processing in Foreground	7-15
The Functions and OK codes	7-16
Session Statuses	7-17
Notes for Processing BI Sessions in Foreground.....	7-18
Session Statistics.....	7-19
Session Log	7-20
Session Analysis	7-21
Session Analysis	7-22
Session Analysis - Screen Field List.....	7-23
Session Analysis - Screen Display	7-24
User Settings in BI Monitor.....	7-25
Exporting and Importing BI Sessions.....	7-26
Import / Export Program Options	7-27
Batch Input Processing: Unit Summary	7-28
Batch-Input Mappen Exercises	7-29
Batch- Input Monitor Solutions	7-30
Legacy System Migration Workbench (LSMW).....	8-1
Course Overview Diagram	8-2
LSMW Basics	8-3
LSMW: Characteristics	8-4
LSMW as an Add-On.....	8-5
LSMW: Core Functions	8-6
The LSMW Concept.....	8-7
LSMW Concept.....	8-8
Generated Programs	8-9

Starting the LSMW	8-11
Project Overview.....	8-12
The Main Steps.....	8-13
Maintain Object Attributes	8-14
Maintain Object Attributes - Standard.....	8-15
Overview of Object Creation.....	8-16
Conversion on Paper.....	8-17
LSMW Structure Definitions	8-18
Flat Source Structure.....	8-19
Source Structures Containing Two Structures.....	8-20
Source Structures With Header and Positions.....	8-21
Maintain Source Structures	8-22
Maintain Source Fields.....	8-23
Field Types	8-24
Maintain Structure Relations.....	8-25
Hierarchical Display	8-26
LSMW Field Mapping and Rules	8-27
Maintain Field Mapping and Conversion Rules	8-28
Field Assignments.....	8-29
Reusability of Rules	8-30
Source Field Assignment	8-31
Rule: Initial.....	8-32
Rule: Constants	8-33
Rule: Transfer.....	8-34
Rule: Fixed Value.....	8-35
Rule: Creating Translation.....	8-36
Rule: Changing Translation.....	8-37
Translation Control Overview.....	8-38
1:1 Values of Conversion Key	8-39
Conversion Value Interval.....	8-40
Rule: Prefix.....	8-41
Rule: Suffix	8-42
Rule: Concatenation.....	8-43
Rule: Transfer Left-Justified	8-44
Rule: ABAP Code	8-45
Rule: Own Routine.....	8-46
Field Mapping and Conversion Rules.....	8-47
LSMW Reading and Transferring Data to R/3.....	8-48
Specify Files	8-49
Source of the Legacy Data.....	8-50
Read Data File	8-51

Wildcards in File Names	8-53
Assign Files	8-54
Read Data	8-55
Display Read Data	8-56
Convert Data	8-57
Display Converted Data	8-58
Starting Transfer	8-59
Starting Standard Transfer Program	8-60
Administrative Functions	8-61
Authorizations	8-62
LSMW: Unit Summary	8-63
LSMW Exercises	8-64
Direct Input	9-1
Course Overview Diagram	9-2
Direct Input Concept	9-3
Direct Input Monitor	9-4
Tasks in the Direct Input Monitor	9-5
Example: FI Documents	9-6
Job Analysis	9-7
Error in FI Document	9-8
Error in Material Master or Sales Document	9-9
Periodic Jobs	9-10
Direct Input: Unit Summary	9-11
Direct-Input Exercises	9-12
The Transaction Recorder: TA Recorder	10-1
Course Overview Diagram	10-2
The TA Recorder	10-3
Overview of Transaction Recorder	10-4
Uses of the TA Recorder	10-5
The Recording	10-6
Changing Customer Data: Initial Screen	10-7
Changing Customer Data: Address	10-8
Results of the Recording	10-9
Special Recording Fields	10-10
Determining the Field Name	10-11
Subscreens	10-12
Tabstrips	10-13
Step Loops and Table Controls	10-14
Recording Guidelines	10-15
Enjoy SAP Controls and Data Transfer	10-16
The Transaction Recorder	10-17

Creating Recording	10-19
Changing Customer Data: Initial Screen.....	10-20
Results	10-21
Assigning Field Names.....	10-22
Editing Recording.....	10-23
Object Type and Import Method: Recording.....	10-24
The Transaction Recorder.....	10-25
Transaction Recorder.....	10-26
Changing Customer Data: Initial Screen.....	10-27
Change Debitor: Payment Transactions (1)	10-28
Changing Customer Data: Payment Transactions (2)	10-29
Hierarchy Display of the Recording.....	10-30
Recording Overview	10-31
Recording Editor.....	10-32
Recording Several Transaction Runs.....	10-33
Editor: Importing and Exporting Recordings	10-34
Processing the Recording.....	10-35
Processing Recording with CT 1	10-36
Processing Recording with CT 2.....	10-37
BI Mode	10-38
Creating a Session From the Recording	10-39
Creating Test Data.....	10-40
Creating a Program.....	10-41
Program Functions.....	10-42
Generating Function Modules	10-43
Function of Function Modules.....	10-44
Import Interface of Function Module.....	10-45
The Transaction Recorder.....	10-46
Generated Program.....	10-47
Structure of the Generated Program.....	10-48
Structure of BDC Table	10-49
Read Structure	10-50
Opening and Closing the File	10-51
Function Modules for BI Progrmm	10-52
Function Module BDC_OPEN_GROUP	10-53
Subprogram OPEN_GROUP	10-54
Function Module BDC_INSERT	10-55
Subprogram BDC_TRANSACTION	10-56
Function Module BDC_CLOSE_GROUP	10-57
Subprogram CLOSE_GROUP	10-58
Subprograms BDC_DYNPRO, BDC_FIELD	10-59

Overview.....	10-61
Call Transaction Program: Example	10-62
CALL TRANSACTION (1)	10-63
CALL TRANSACTION (2)	10-64
CALL TRANSACTION (3)	10-65
CALL TRANSACTION (4).....	10-66
Structure CTU_PARAMS.....	10-67
Return Code and System Fields	10-68
Generated Program of the TA Recorder.....	10-69
The Transaction Recorder.....	10-70
Integrating into the DX-WB	10-71
DXWB Release 4.5	10-72
Creating Program as DX Object	10-73
Advantages	10-74
The Transaction Recorder: Unit Summary	10-75
Transaction Recorder Exercise 1	10-76
Transaction Recorder Exercise 2	10-78
Appendix: Include BDCRECX1.....	10-80
Special Methods	11-1
Course Overview Diagram	11-2
Special Methods: Contents (1)	11-3
BI and Interactive Lists	11-4
Transaction Recorder Results	11-5
Editing the Recording	11-6
Special Methods: Contents (2)	11-7
External Data in Variable Data Format	11-8
Reading and Processing Variable Format	11-9
Special Methods: Contents (3)	11-10
Synchronous Processing.....	11-11
Asynchronous Processing.....	11-12
Special Methods: Contents (4)	11-13
The New Selection Screen for ZRFBIDE00.....	11-14
Modifying ZRFBIDE00 for Call Transaction (CT).....	11-15
Special Methods: Contents (5)	11-16
SAP Notes on Data Transfer	11-17
BI Utility Programs	11-18
BI Programming: Reduce Rollback Segment Load.....	11-19
Further Tips and Information	11-20
Special Methods: Unit Summary	11-21
Special Methods Exercises	11-22
Special techniques Solutions	11-23

Course Overview Diagram	12-2
Background Processing - Contents (1)	12-3
Concept of Background Processing	12-4
Background Processing Steps	12-5
Creating a Job (1)	12-6
Creating a Job (2)	12-7
Creating a Job (3)	12-8
Creating a Job (4)	12-9
Job Overview (1)	12-10
Job Overview (2)	12-11
Background Processing - Contents (2)	12-12
Parallel Processing with RSBDCSUB	12-13
Batch Input Scheduling with RSBDCSUB	12-14
Jobs	12-15
RSBDCSUB Jobs	12-16
Frontend Access in Background	12-17
Background Processing: Unit Summary	12-18
Background Processing Exercises	12-19
Background Processing Solutions	12-22



- R/3 System
- Release 4.6B
- May 2000
- Material number 50039585

Copyright 2000 SAP AG. All rights reserved.

Neither this training manual nor any part thereof may be copied or reproduced in any form or by any means, or translated into another language, without the prior consent of SAP AG. The information contained in this document is subject to change and supplement without prior notice.

All rights reserved.

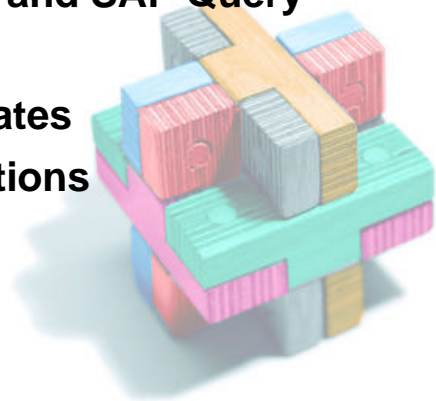
© SAP AG 1999

■ Trademarks:

- Microsoft ®, Windows ®, NT ®, PowerPoint ®, WinWord ®, Excel ®, Project ®, SQL-Server ®, Multimedia Viewer ®, Video for Windows ®, Internet Explorer ®, NetShow ®, and HTML Help ® are registered trademarks of Microsoft Corporation.
- Lotus ScreenCam ® is a registered trademark of Lotus Development Corporation.
- Vivo ® and VivoActive ® are registered trademarks of RealNetworks, Inc.
- ARIS Toolset ® is a registered Trademark of IDS Prof. Scheer GmbH, Saarbrücken
- Adobe ® and Acrobat ® are registered trademarks of Adobe Systems Inc.
- TouchSend Index ® is a registered trademark of TouchSend Corporation.
- Visio ® is a registered trademark of Visio Corporation.
- IBM ®, OS/2 ®, DB2/6000 ® and AIX ® are a registered trademark of IBM Corporation.
- Indeo ® is a registered trademark of Intel Corporation.
- Netscape Navigator ®, and Netscape Communicator ® are registered trademarks of Netscape Communications, Inc.
- OSF/Motif ® is a registered trademark of Open Software Foundation.
- ORACLE ® is a registered trademark of ORACLE Corporation, California, USA.
- INFORMIX ®-OnLine for SAP is a registered trademark of Informix Software Incorporated.
- UNIX ® and X/Open ® are registered trademarks of SCO Santa Cruz Operation.
- ADABAS ® is a registered trademark of Software AG

- The following are trademarks or registered trademarks of SAP AG; ABAP/4, InterSAP, RIVA, R/2, R/3, R/3 Retail, SAP (Word), SAPaccess, SAPfile, SAPfind, SAPmail, SAPoffice, SAPscript, SAPtime, SAPtronic, SAP-EDI, SAP EarlyWatch, SAP ArchiveLink, SAP Business Workflow, and ALE/WEB. The SAP logo and all other SAP products, services, logos, or brand names included herein are also trademarks or registered trademarks of SAP AG.
- Other products, services, logos, or brand names included herein are trademarks or registered trademarks of their respective owners.

- Section **Basis Technology Overview**
- Section **ABAP Workbench Concepts and Tools**
- Section **Managing ABAP Development Projects**
- Section **ABAP Dictionary**
- Section **ABAP Programming Techniques**
- Section **Techniques for List Creation and SAP Query**
- Section **Transaction Programming**
- Section **Programming Database Updates**
- Section **Enhancements and Modifications**
- Section **Data Transfer**

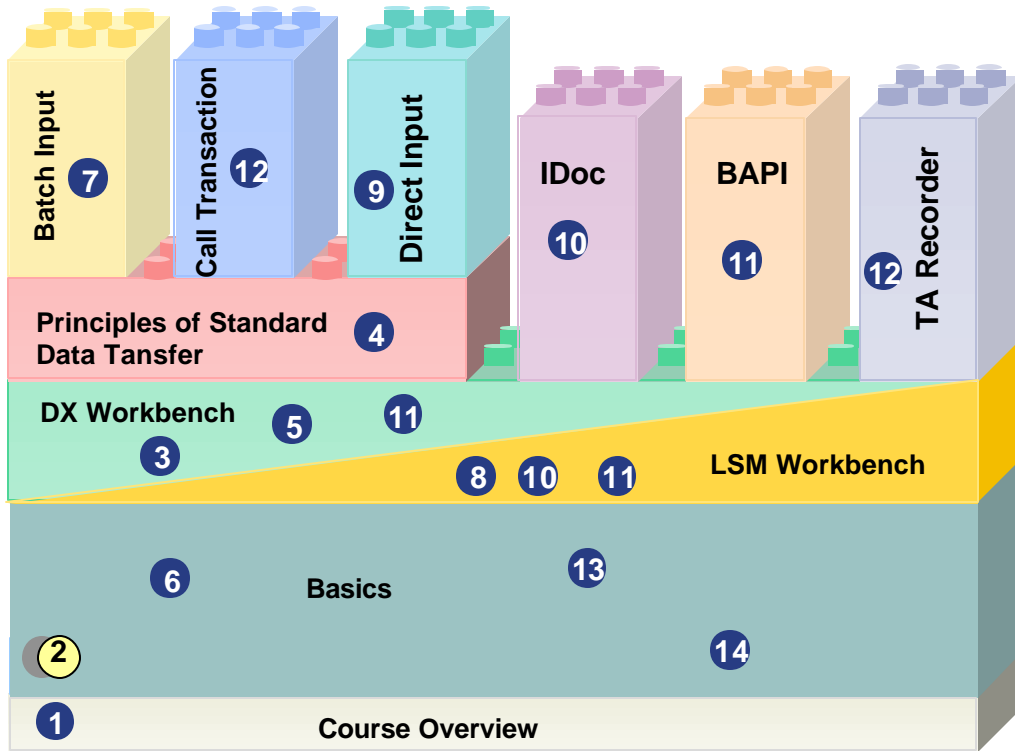




Unit	Data Transfer	Unit	Legacy System Migration Workbench (LSMW)
Unit	Project Concept	Unit	Direct Input
Unit	Principles of Standard Transfer	Unit	TA Recorder
Unit	DX-WB Tools	Unit	Special Methods
Unit	Sequential Files	Unit	Background Processing
Unit	Batch Input Processing		

- **Task**
- **Solutions**

Course Overview Diagram

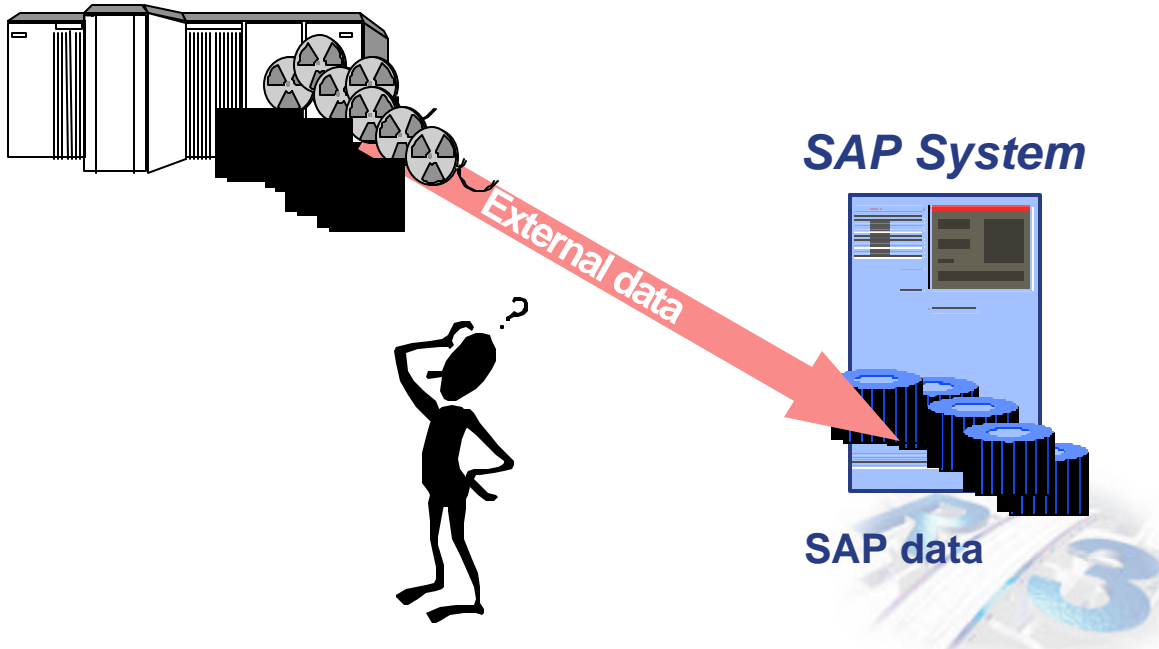




Task

Solutions

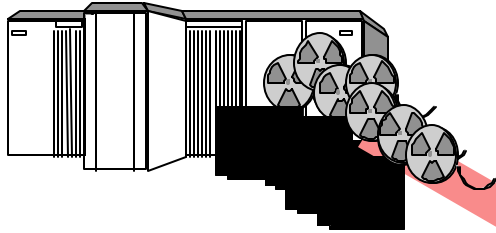
External system



© SAP AG 1999

- For reasons of efficiency, large volumes of data cannot be transferred manually from an external system into the R/3 System. A data transfer is required that transfers the data automatically in the background.
- A transfer is required, for example, when:
 - A new SAP System is installed and data has to be transferred from another system.
 - Data is regularly transferred from external systems into the SAP System.
Example: if the data in some areas of the company is created in external systems and this data has to be integrated into the SAP System.
- A way has to be found to transport the data into the R/3 System.

External system



SAP System



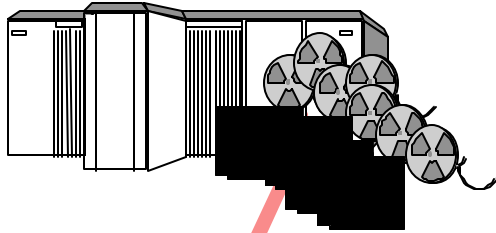
SAP data



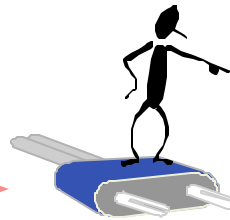
© SAP AG 1999

- External data cannot be directly imported into the R/3 database because data integrity is not guaranteed.

External system

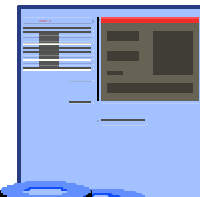


**Sequential
file**



**SAP
interfaces/
checks**

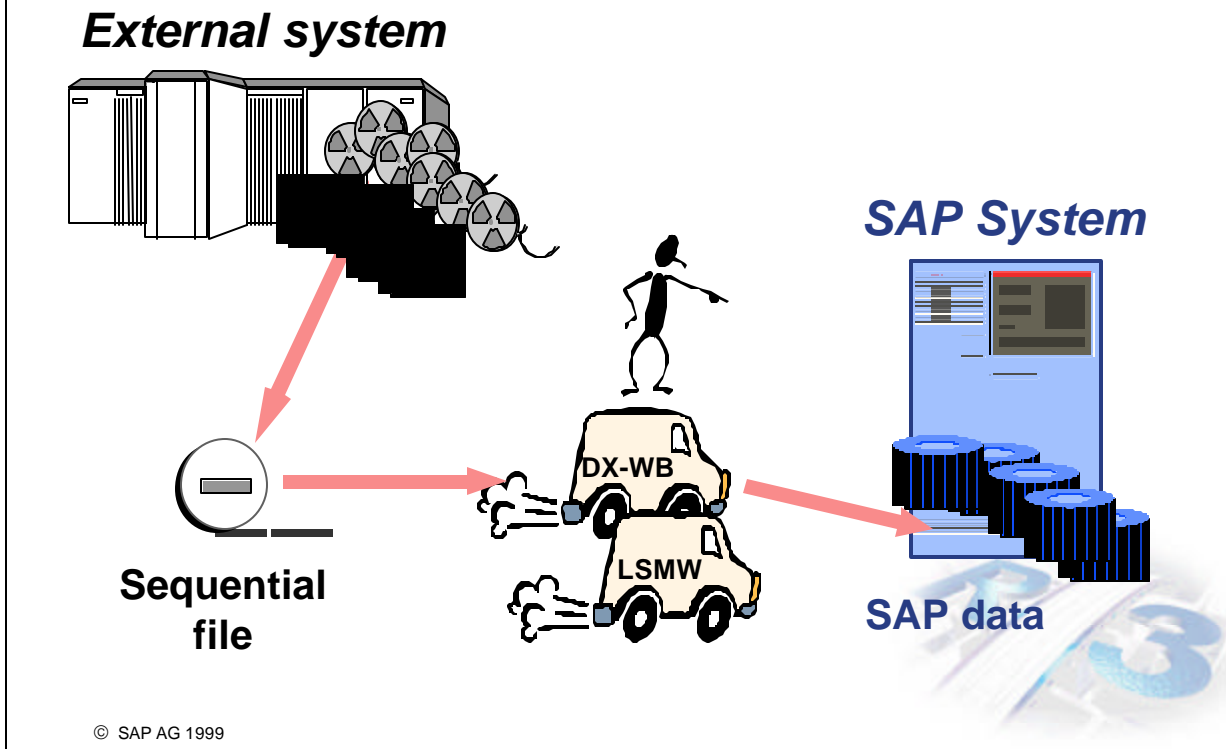
SAP System



SAP data

© SAP AG 1999

- Data integrity has to be guaranteed when transferring data from another SAP System or from an external system.
- The same checks that are used online have to be carried out for the data transfer.
- As the online checks are very extensive in the transactions and are partly cross-application, it is very difficult to carry these checks out yourself.
- SAP provides special interfaces for data transfer. These partly use the transactions and their checks to transfer data into the SAP System.



- The following tools are provided for transferring data from another SAP System or from an external system into the SAP System:
 - The Data Transfer Workbench (DX-WB).
 - The Legacy System Migration Workbench (LSMW).
- These tools cannot transfer data themselves. They are used as a central access point for the transfer and they make it easier by using the actual transfer programs of the applications.



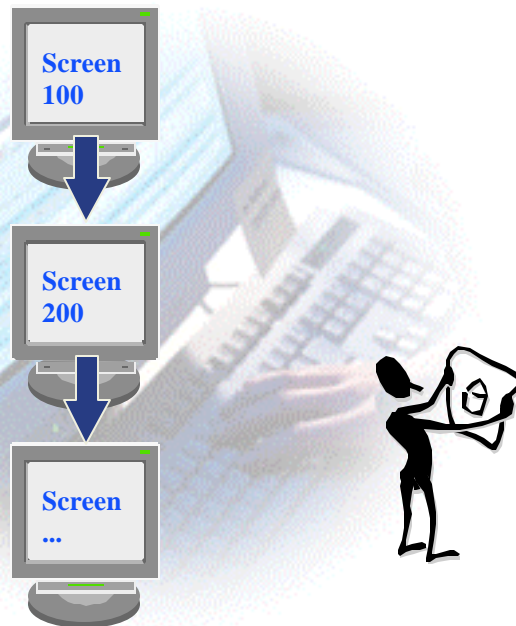
Task

Solutions



Online

SAP Transaction



© SAP AG 1999

- Run through the transaction used to transfer the data to the SAP System in a dialog operation.
- Using transaction analysis, determine the following information:
 - The transaction code, if unknown
 - The fields where input is required
 - The fields where default values can be used
 - The names, types, and lengths of the fields used in the transaction

- One of the most important tasks in data transfer is to specify which data fields of the external system correspond to which data fields in the SAP System.
- Method: First examine the structure of the data that is to be transferred into the system. Determine how the existing data is to be mapped to the SAP data structure.
- Determine the fields that can be transferred directly from the existing data. With these fields there is a direct correspondence between the existing data field and the associated SAP data field.
- Determine the fields of the external system that have to be modified so that they can be transferred into the SAP System.
- Check whether some fields will need conversion of their data types and/or data lengths.

- Most of the data from the external system must be converted into SAP format. We call this **“formatting”** the data. Often the valid value set in SAP data fields have to be determined and, if necessary, convert the external system values to the the new valid SAP value set.

- In this course the icons shown above are used for formatting and assignment. If formatting and assignment (also known as mapping) are done in one step, then it is called **conversion**.

- To transfer external data into the SAP System you have to convert the data into SAP format. The data can be converted in various ways:
 - *In the external system* - this has the advantage that the user knows the programming language (e.g. COBOL) used in the system.
 - *In the SAP System* with ABAP - this has the advantage of automatic type conversion. The structure definitions of the ABAP Dictionary can be used directly.
 - *With Excel* - simple and complex conversions of flat structures are possible.
 - With the *LSMW* mapping tool - the advantages of R/3 plus an automated mapping tool
 - With a certified *external product*

- The DX-WB cannot carry out the data transfer itself. It is used as a framework that makes the actual data transfer easier.
- The DX-WB supports the following transfer methods:
 - Standard transfer programs
 - Transfer using BAPIs
 - Transaction recorder

- Rules, defined in the LSMW are used to convert the data from the external system.
- Like the DX-WB the LSMW cannot actually transfer the data. The LSMW is used to convert the data and call the appropriate data transfer program in the application.

- The LSMW supports the following transfer and import technologies:
 - Standard transfer programs used in applications
 - IDoc and BAPI technology
 - Recording function of the transaction recorder

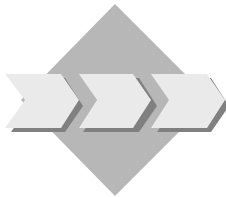


Unit: Overview of Data Transfer

Topic: Getting to know SAP Applications for the Data Transfer to be executed



- Getting to know the SAP transaction for entering the debtor data online.



Debtor data of a legacy system are to be entered online in the R/3 system.



Note: (## is the group number)

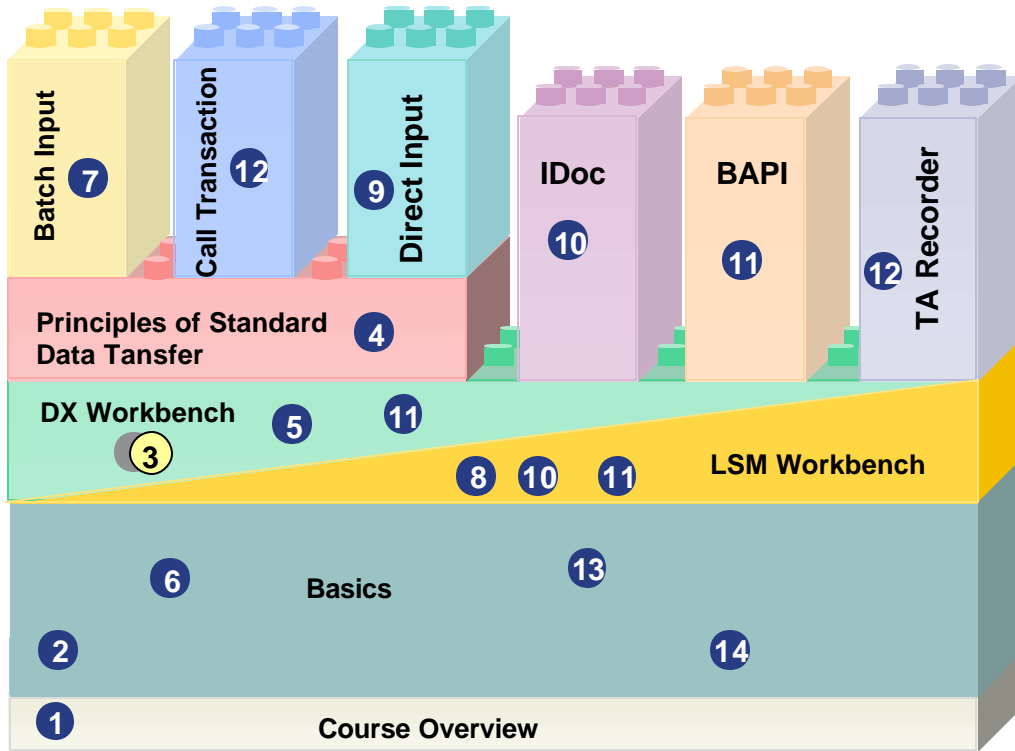
Debtors : Z-##-00001 and Z-##-00002

- 1 For the external data transfer of debtors, analyze the SAP transaction FD01 (Create debtors: Initial screen accounting).
 - 1-1 Create the debtors using the numbers “Z-##-00001” and “Z-##-00002”. Fill all obligatory fields with optional, appropriate values.
 - 1-2 Initial screen: Create the debtors with the account group “customers general” respectively “KUNA” with external number assignment in company code “0001”.
 - 1-3 Address: Maintain the address data. For the language key and country identification you can use “GB” or “DE”.
 - 1-4 Control data: Maintain the VAT registration number (VAT reg. no.) with the value DE123456789.
 - 1-5 Switch to the company code data. In the accounting information: maintain the reconciliation account with the value “120000”.
 - 1-7 Save the debtor.
- 2 You can check the debtors you created using SAP transaction FD03 (Display debtors: Initial screen accounting).

Tip: The area code on the second screen of the transaction must be filled with data, too.

- **Transfer tools and methods**
- **Tasks in data transfer projects**

Course Overview Diagram





Technologies and Tools

Project Overview

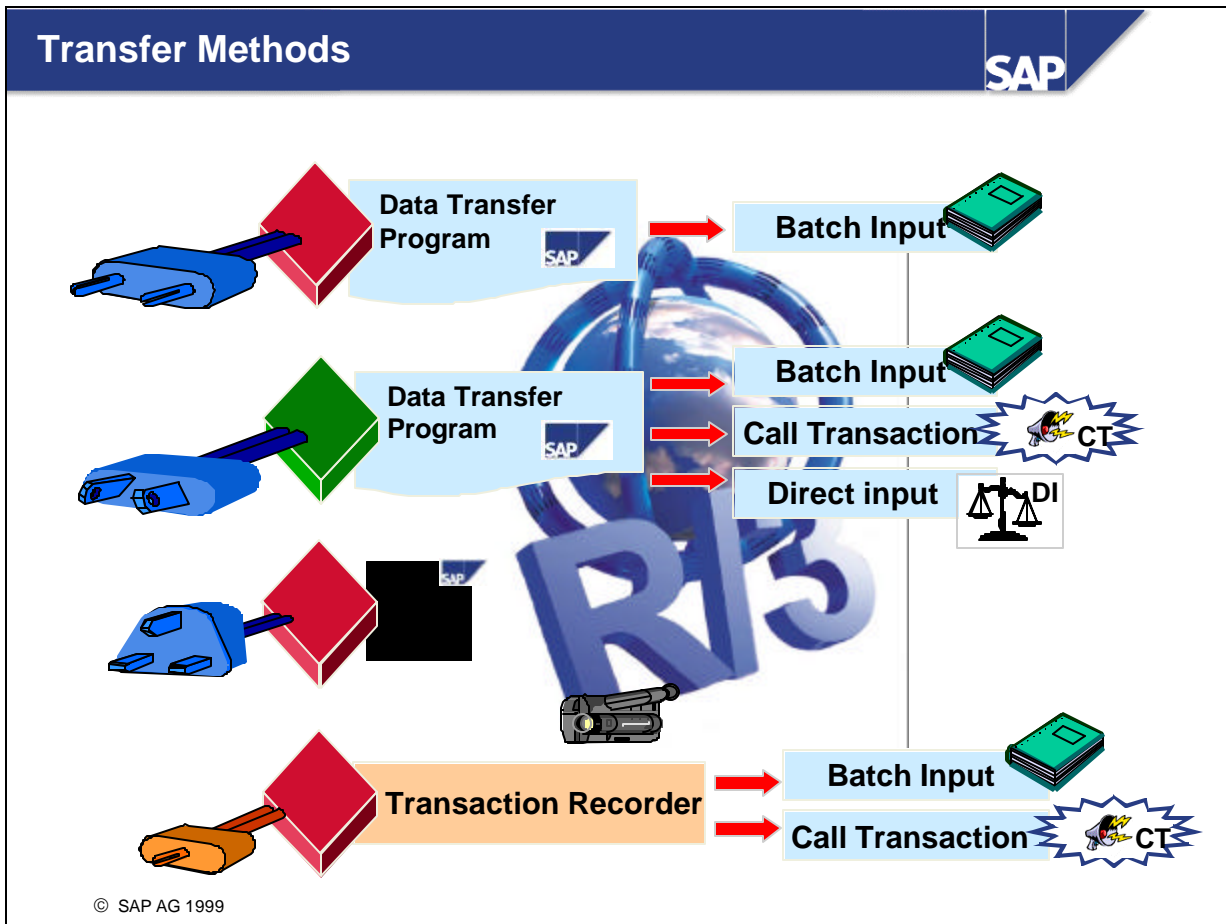
Business Objects

Project in the DX-WB

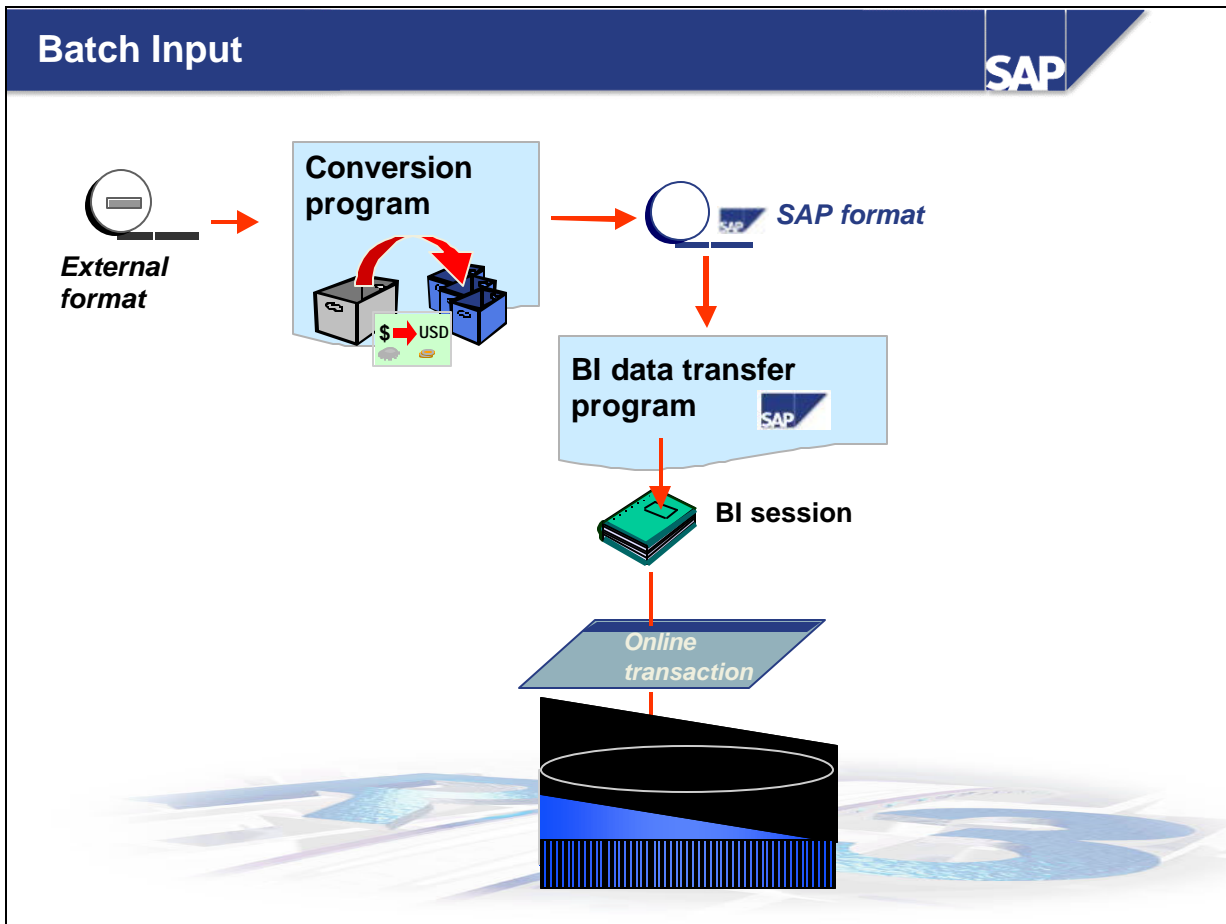


© SAP AG 1999

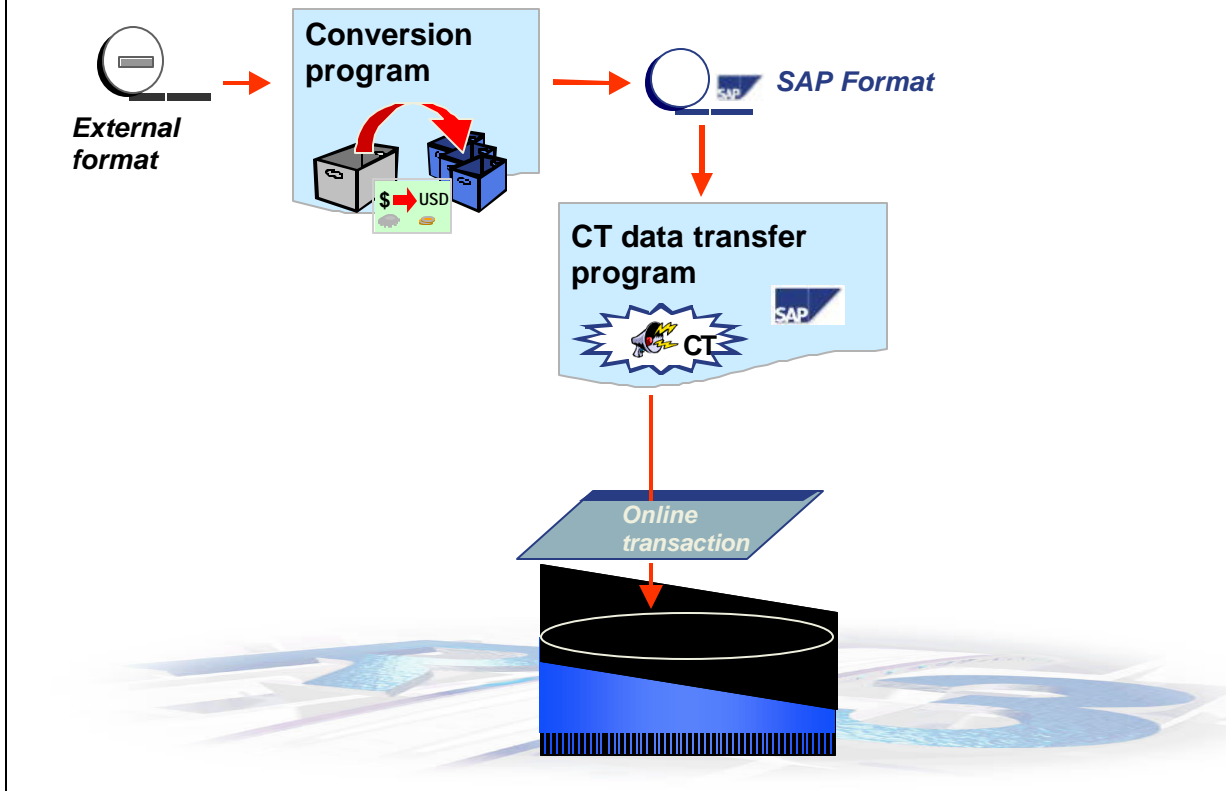
- Each application provides its own transfer program. As other data always plays a role in each application, the interfaces are not the same.



- Most applications provide a standard interface.
- Several data transfer methods are supported in the SAP System. These are batch input, call transaction, direct input and BAPIs. Which method is supported in a certain situation depends solely on the application involved. The expected amount of data usually plays a decisive role in which methods are supported.
- In special cases and for a few applications, a standard interface may not be provided. If this is the case, you can use the transaction recorder (TA) instead. The TA supports both batch input and call transaction.



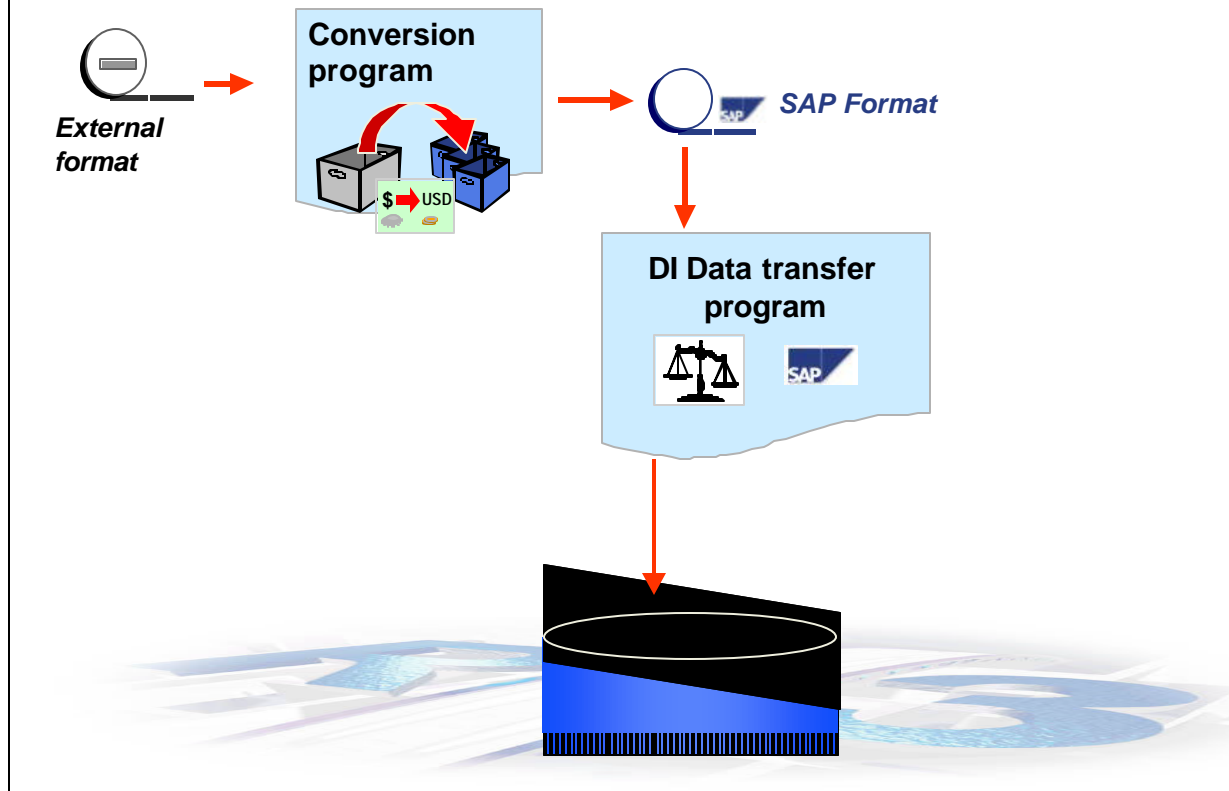
- Batch input is a standard technique used to transfer data into the R/3 System. The transaction process is simulated and the data is transferred as if it had been entered online. The advantage of this process is that all the transaction checks are carried out and therefore data consistency is assured.
- The batch input process runs in two phases:
 - 1.) A batch input session is created containing all the relevant data.
 - 2.) The batch input session is processed and the data contained in it is imported into the R/3 System.
- Most of the SAP standard data transfer programs use batch input.
- Note: The online transaction is used for importing, checking and transferring the data into the R/3 database.



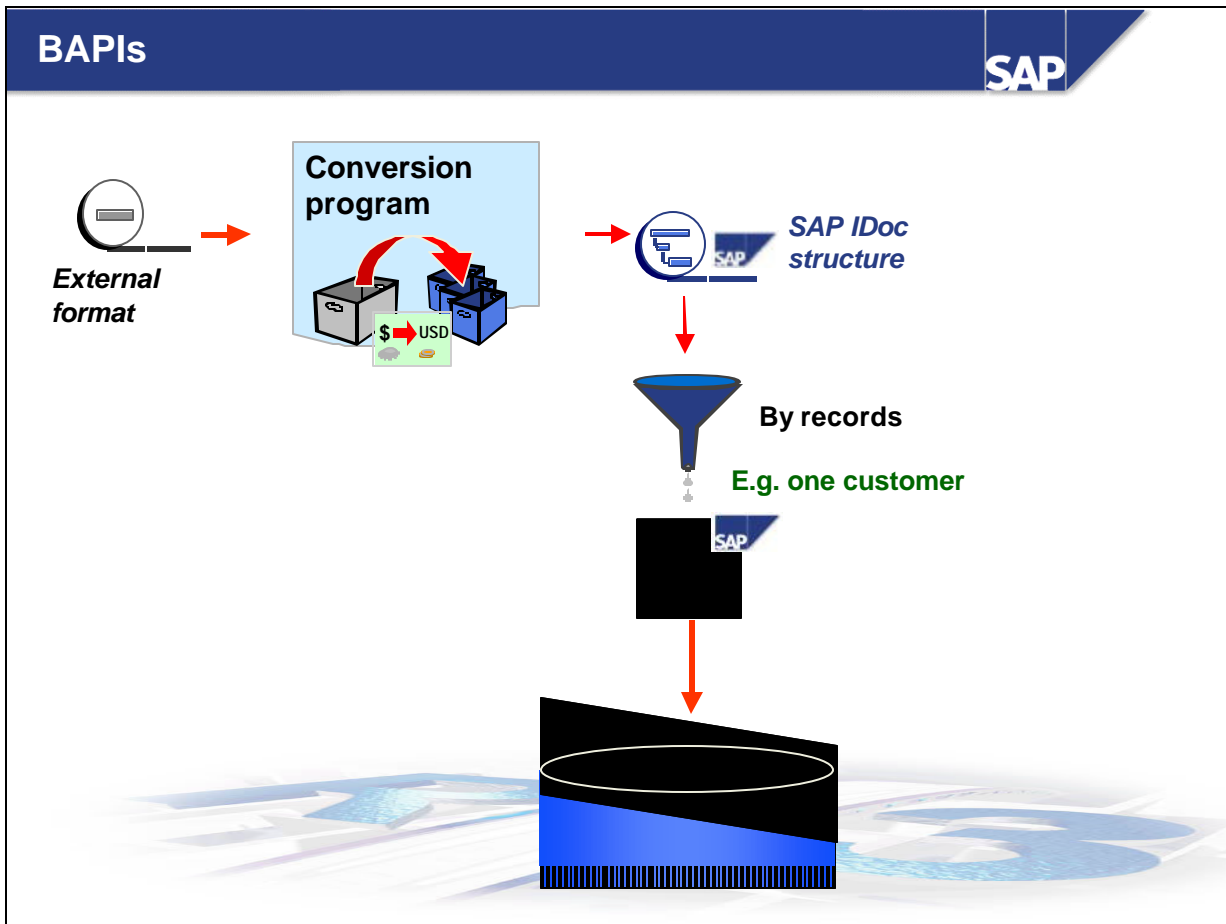
- Call transaction is a standard technique used to transfer data into the R/3 System. The transaction flow is simulated and the data is transferred as if it had been entered online. The advantage of this procedure is that all transaction checks are carried out, thereby guaranteeing data consistency.
- Call transaction posts the data directly via the online transaction.
- This method provides the same functions as with batch input.
- Note: the online transaction is used for importing and checking.

Direct Input Technique

SAP



- With direct input the data in the data transfer file is checked thoroughly and then transferred directly into the R/3 System. The R/3 database tables are loaded/updated with the new data.
- The direct input program used for importing and checking the data carries out the same tests as the online transaction. This guarantees data integrity.
- The direct input method uses function calls in place of the online transaction screens.
- Note: the direct input method is not available for all data transfer applications.



- BAPIs are standardized programming interfaces that provide external access to the business processes and data of the R/3 System.
- Business Application Programming Interfaces (BAPIs) are defined in the Business Object Repository (BOR) as methods of SAP business objects or SAP interface types and enable an object-oriented view of the business components (application components) of the R/3 System.
- The data is transferred by calling a BAPI in the application.
- The same checks are made as with online transfer.

- You use the transaction recorder to create a recording of a sequence of transactions along with their corresponding sequence of screens.
- You can generate a data transfer program from the recording; this program will be used to import the data into the R/3 System. The program can execute data transfer using either the batch input or call transaction methods.

- If a data transfer is to be carried out, it should be implemented in a project for each application. Projects can be divided into sub-projects as needed.
- A project team should be assembled for the data transfer. Each member of the project should be able to carry out specified tasks of the entire project, so that the team is in a position to be able to carry out the complete process.
- After forming the team the tasks involved in the project should be specified.
- After the tasks are specified, they should be assigned to team members. One person may have several roles.

- Customizing is extremely important for data transfer. Customizing changes automatically result in changes to the checks and valid values of the R/3 System. For data transfer this means that data that could be transferred error-free before Customizing was changed, may be transferred with errors after changes in Customizing. Therefore, always test data transfer after every Customizing change.
- To execute the data transfer, you must have the required background knowledge and understand the techniques for extracting data from external systems, as well as the procedures for transferring the data to the SAP System.
- Note: You must also have comprehensive knowledge of the applications whose data is to be transferred.
- Another question to be answered is: Are there enough people on the team to carry out the required tasks?

- The tasks listed above should be executed in a data transfer project.

- A test should always be run with a representative amount of data. This means the test data should include as many **typical data records from the external system** as possible. Testing using 100 data records that are all copies of the first is of no use at all.
- There are three phases in the test:
 - Converting the data into the interface format
 - Transferring the data to the R/3 System
 - Testing the data in the R/3 System
- If data errors are found, fix these errors in the legacy system. Data quality is extremely important for data transfer. Data records with errors usually result in errors in the data transfer.

Therefore, the data from the legacy system must be 100% correct to ensure error-free data transfer.

- The appendix has additional information on the test procedure.

- BOR: Business Object Repository
 - Central repository within the R/3 System containing all the SAP business object types, SAP interface types, their definitions and methods etc.
 - It identifies and describes the available SAP business object types and interface types and their BAPIs.
 - For example, if you are developing an application program you use the key fields and the BAPI methods from the information stored in the BOR on the SAP business object types or SAP interface types. The BOR contains all the details you need to integrate the correct object type definitions and BAPI calls into your application program.
 - Likewise, all the transfer programs as of 4.6 are organized by the BOR objects in the application.

- **BAPI: Business Application Programming Interface**
 - Standardized programming interface that enables access to business processes and data in the R/3 System.
 - BAPIs are defined in the BOR as **methods** of SAP **business object types** that carry out specific business functions.
- There may be different transfer methods for each BOR object. These may be BAPIs or transfer programs. As the BAPIs are already defined through BOR objects, the standard transfer programs are also closely defined through them. So a BOR object can identify all the transfer programs that it can use.
- The BOR is the R/3 interface to external systems, so all the transfer programs have been assigned to the BOR objects.

- The Data Transfer Workbench provides you with integrated project management to carry out all the steps required for transferring data into your R/3 System.
- The DX-WB provides various tools to support you with the data transfer.
- SAP provides a range of standard data transfer programs and BAPIs for loading data into R/3. These can be called from the DX-WB.

- The functions of the DX-WB are explained on following slides using the transfer of customer data.
- The prerequisite is that the external data is available in the file system of the R/3 application server.
- The following tasks are to be carried out:
 - Format data
 - Map data to the target structure
 - Transfer data into the SAP System

- SAP provides standard data transfer programs for most of the business objects and BAPIs. These are registered as task type LOA (load).
- If you need other programs for the data transfer, you have to create them. You have to register them before you can use them in the DX-WB .
- In the example above, each of the programs (A,B and C) must be registered in one task type.
- Note: In the standard system a batch input program is registered in task type LOA for loading customer data . In the example however, we will register our own load program that copies the data to R/3 with call transaction because we do not yet want to bother with creating and processing batch input sessions.
-

- Each program, function module and BAPI that you register must be assigned to a BOR object. The BOR object will be available to them later.
- Each program, function module and BAPI must be assigned to a task type. The task type will be available to them later.

- You create programs and function modules for tasks such as extracting data from the external system or cleaning up this data, and register them with the corresponding object type.
- In the data transfer workbench, choose *Goto ® Registration* and the relevant option:
 - *Programs and function modules*
 - Choose *Create registration*.
 - In the dialog box, choose values using *Possible entries* help:
 - *Object type*
 - Choose an object type.
 - *Task type* and *Program type*
 - Each task type is assigned several program types.
- Registered programs are available in the Repository for other users.

■ You create a transfer project in the DX-WB, where you can create, change, display, and delete:

- Projects
- Subprojects
- Runs
- Tasks

■ The following pages describe each of these steps in detail.

- Projects are used to group the transfer of business objects.
- For **each business object type** you want to transfer, you must create at least one **subproject**. For each subproject, you must create at least one flow definition and one task.
- Each project name must be unique within a system, as the name is used as a key.

- Subprojects are used to subdivide projects. **Each subproject is assigned exactly one business object type** whose data is transferred as part of that subproject. Within a project, you can create several subprojects for the same business object type (for example, subprojects for *Document data transfer from plants 1, 2, and 3*).
- If after reading all the descriptions of the available objects you cannot find the object you need, see the application documentation for more information.
- Subproject names in the system must be unique as they are used as a key.

- In the run specification you specify **how the data for a subproject is transferred**, by creating tasks and specifying the files (see *Task (n)*). You can create more than one run definition for each subproject. Depending on the type different methods are possible for each run.
- You can transfer the data in parallel by creating several run definitions with different file names in one subproject and then starting parallel runs.

- For each object type that you select you have to select a task type (extract, map, load...).
- Program types are assigned to each task type (BAPI, function module, batch input...).
- One or more program types are provided by SAP to load the data into the R/3 System (task type LOA) :
 - **LOA** **BAPI** Loads data into R/3 using a BAPI
 - BINP** Loads data into R/3 using Batch-Input
 - DINP** Loads data into R/3 using direct input
 - FUNC** Loads data into R/3 using a function module
 - REPO** Loads data into R/3 using a report

- The task type describes "what" is done, the program type describes "how" it is done.

- When you call programs, you have the option to transfer the file name to the program using a default interface.
- To do this you have to specify the file name in the popup window. You should only do this, if the program supports this interface.
- You can find information about the interface in the application help documentation for the DX-WB under *Programs, Developing function modules and BAPIs*.
- Depending on the program type the attributes may look different.

- The actual data transfer from a file is executed during a run. When the run is started, the system starts all the tasks in the flow definition by calling the corresponding programs.
- Tasks are automatically executed in order from top to bottom.
- The run stops if there are errors in a task.
- You can execute any number of runs for a flow definition. Each run has a status and a log (application log). If needed, you can restart a run that has not completed.
- After a task of program type REPO, BINP, or DINP has run, you must define the status. The status of all other tasks is set automatically by the task.
- The following statuses are available for every started run in a flow definition:
 - Task terminated,
 - Task completed without errors
 - Task completed with errors, no restart, and task manually ended.

- A log is generated for every executed run in a flow definition:
- To view the log, choose *Display log*.
- The log contains all actions that occur during a project. Every task executed within a run can create messages in this log. The project log itself is contained in the application log.

- If a run did not complete successfully, you can choose from the possible actions:
 - Continue run:

To complete a run that did not finish successfully, fix the error using the log as a source of error information, position the cursor on the run ID (for example, 003), and choose *Continue run*.
 - Cancel run:

If a run cannot be completed, cancel the run by choosing *Cancel run*.

This run must be canceled before any other run can be executed.
 - Cancel task:

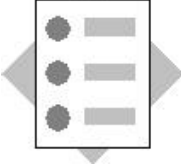
If a task has not been completed, you can still set it to completed by choosing *Cancel task*. When the run is continued, this task will be skipped.
 - Delete run:

You can delete runs that have completed or been manually terminated, by choosing *Delete run*.

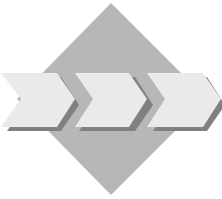


Unit: Project Concept

Topic: Create Data Transfer Project in DX-WB



- Creating a transfer project in the DX-WB and using it to transfer debtor data in the R/3 system. To do this you have to create a project, a subproject, a flow and three tasks in the data transfer workbench.



Debtor data from a legacy system will be imported in the R/3 system.



Program template: SAPBC420_PRPD_FORMAT_DEBI

Conversion program: ZBC420_##_FORMAT_DEBI

Project: FI-##

Subproject: DEB-##

Run: CT-##

Tasks: IMP-##, MAP-##, CON-##

Load program: SAPBC420_PRPD_RFBIDE00_CT

1 A project is to be created in the data transfer workbench. This project consists of three tasks. You have to register the following for the conversion program task:

1-1 Create the development class ZBC420_##.

1-2 Copy the program SAPBC420_PRPD_FORMAT_DEBI to the name ZBC420_##_FORMAT_DEBI and **generate the program!**

1-3 Register the program ZBC420_##_FORMAT_DEBI in DX-WB as follows:

- Object type: KNA1 (Debtor)
- Task type: MSC (Miscellaneous)
- Program type: REPO (Report)
- Program: ZBC420_##_FORMAT_DEBI

2 Create a **transfer project**.

- 2-1 Create the project *FI-##* with the description *FI group-##* in the DX-WB.
- 2-2 Under the project, create the subproject *DEB-##* with the description *Debtors* and the objekt type KNA1.
- 2-3 Create the run *CT-##* with the description *Transfer with CT* under the subproject.
- 2-4 Create the following three tasks. The tasks portray the following process steps:
Format data, map data and transfer data.

2-4-1 Create the following task with the run *CT-##*:

- Task: FOR-##
- Description: Format
- Task type: MSC
- Program type: REPO

Select the program ZBC420_##_FORMAT_DEBI.

2-4-2 Create the following task in the run *CT-##*:

- Task: MAP-##
- Description: Map
- Task type: MAP
- Program type: REPO

Select the program SAPBC420_PRPD_SAP_REC_DEBI_IN2
Define the file name from the output file:
File type=P
File name= BC420_##_DEBI.SAP

2-4-3 Create the following task in the run *CT-##*:

- Task: IMP-##
- Description: Transfer
- Task type: LOA
- Program type: REPO

Select the program SAPBC420_PRPD_RFBIDE00_CT.

Maintain the file name of the input file:

File type = P

File name = BC420_##_DEBI.SAP

- 2-4-4 Please note that you need to correct the order of the tasks. **Format data, map data and transfer data.**

2-5 Start the run and the corresponding programs. Confirm all tasks manually.

2-6 The debtors are created with an internal number assignment. Write down the debtor numbers:

Debtor 1: _____

Debtor 2: _____

Debtor 3: _____

Debtor 4: _____

Debtor 5: _____

Debtor 6: _____

Debtor 7: _____

Debtor 8: _____

Debtor 9: _____

Debtor 10: _____

2-7 Take a look at the log!

3 Check if the debtors have been created using transaction FD03.



Chapter: Project concept

- 1-3 Go to the registration menu from the DX WB with *Goto-> Registration -> Programs and function modules*. Register the program.

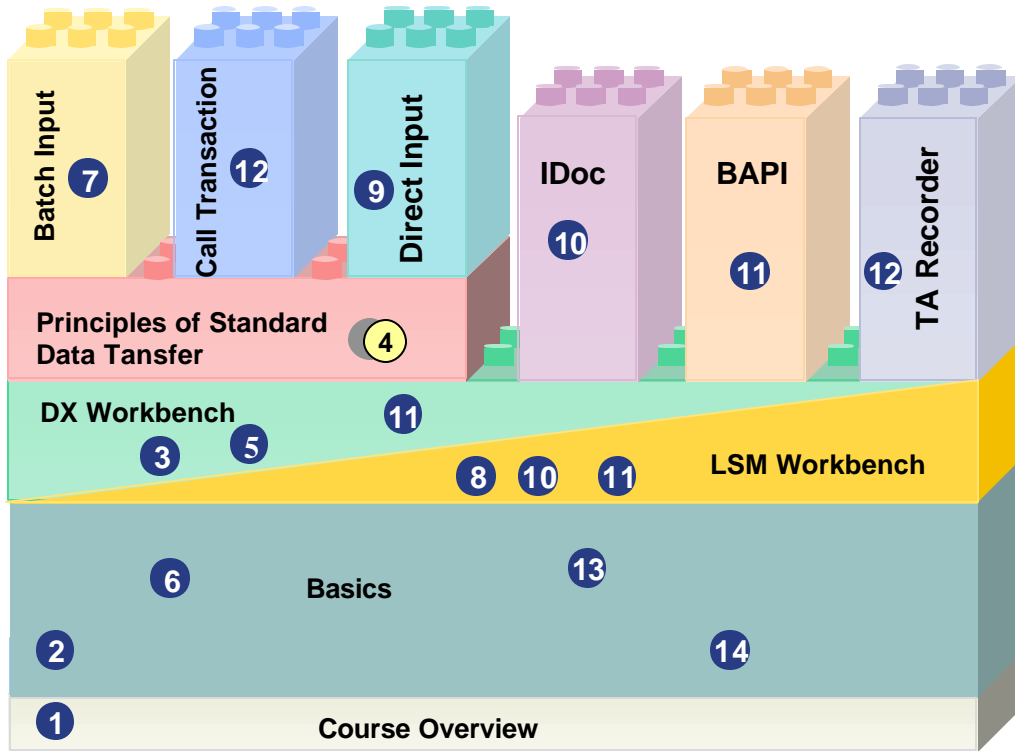
- 2 Go to the DX WB initial screen and create the project, subproject and the run.
 - 2-4 Create tasks according to the task descriptions.

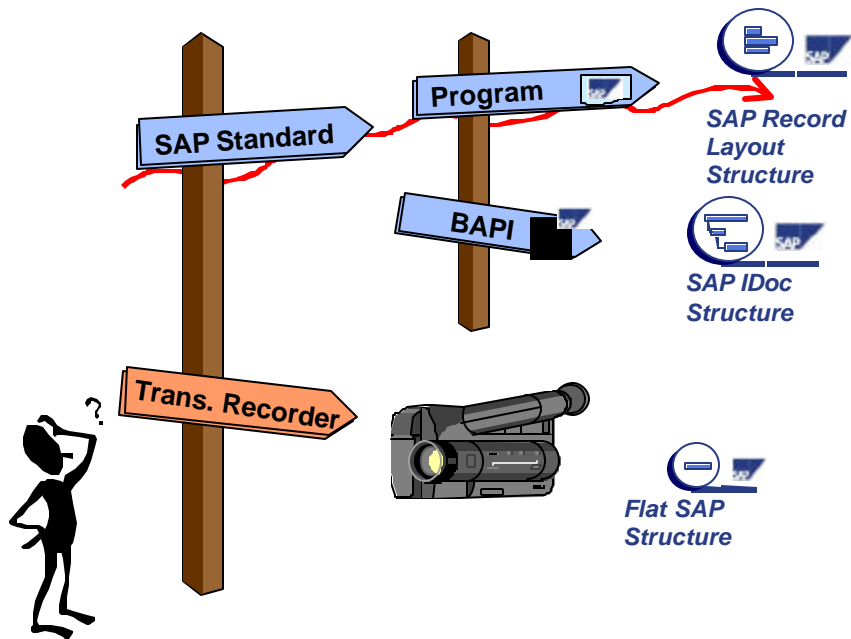
Note: Change the task sequence as appropriate.
 - 2-5 Start the run. All programs are executed consecutively. You must run the program in the selection screen, end it with *Back* and confirm the status manually.

Contents:

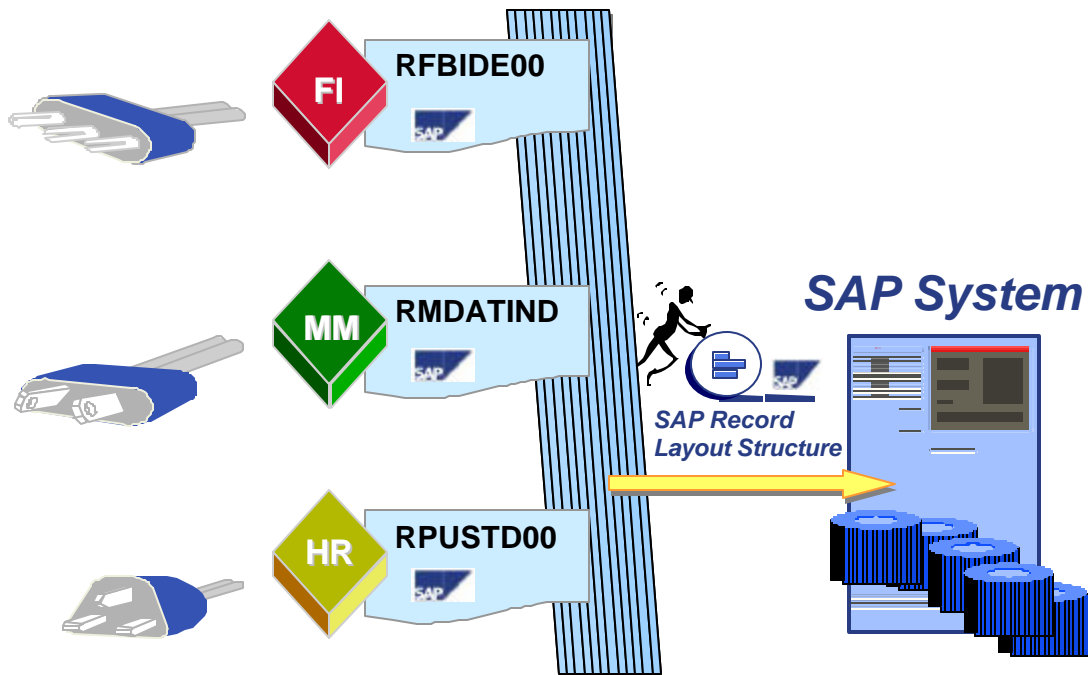
- **Tasks of standard data transfer**
- **Documentation for data transfer**
- **SAP record layout structure**

Course Overview Diagram





- The following describes the basics of data transfer in the standard system using the transfer programs delivered from SAP.

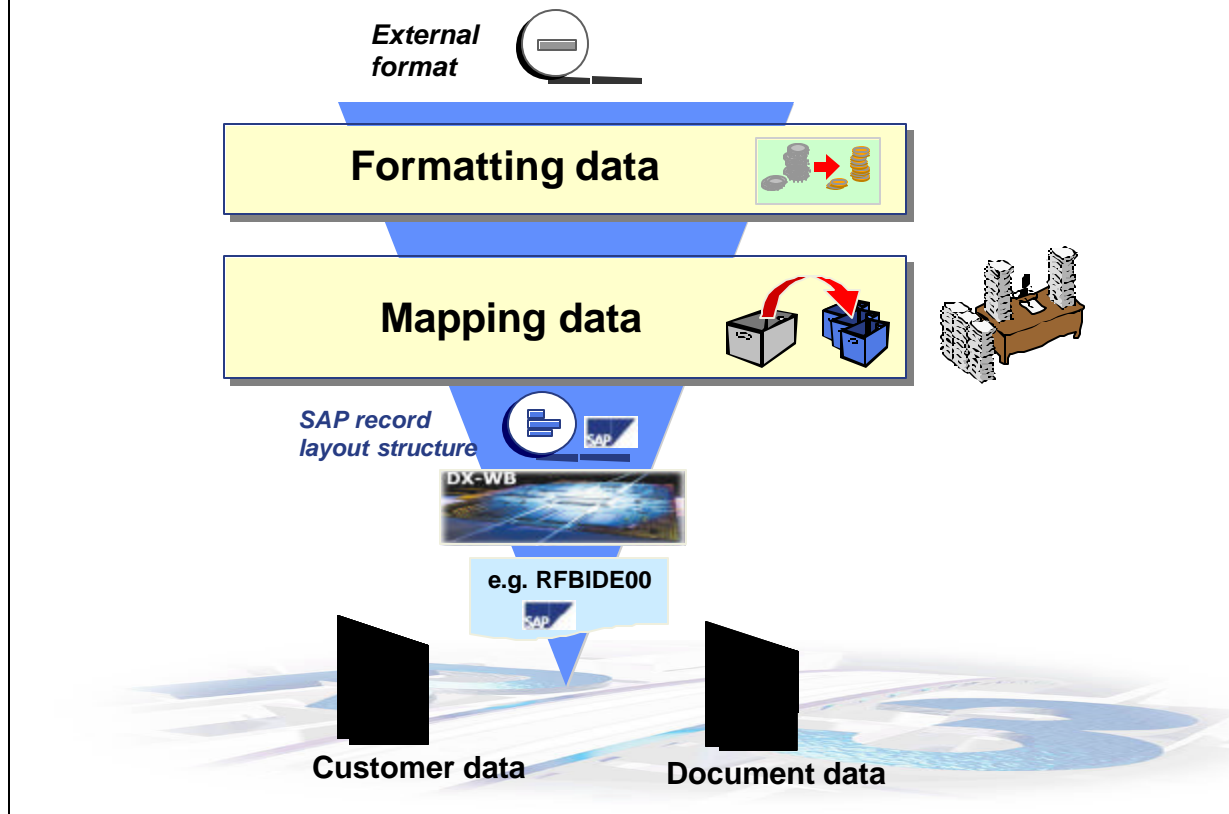


© SAP AG 1999

- SAP supplies standard data transfer programs that require specific SAP-configured interfaces called SAP record layout structures. The external data transferred through these layout structures must be in a compatible format.
- The interface definitions are set in the SAP System and can be used as the basis for projection and planning of the transfer procedure.

Tasks in Standard Transfer

SAP



- The slide shows the important steps in a standard data transfer :
 - The external data is reformatted into the SAP format
 - The external fields are assigned to the SAP record layout fields
 - With DX-WB the SAP standard process can be selected and the data transfer started

Help



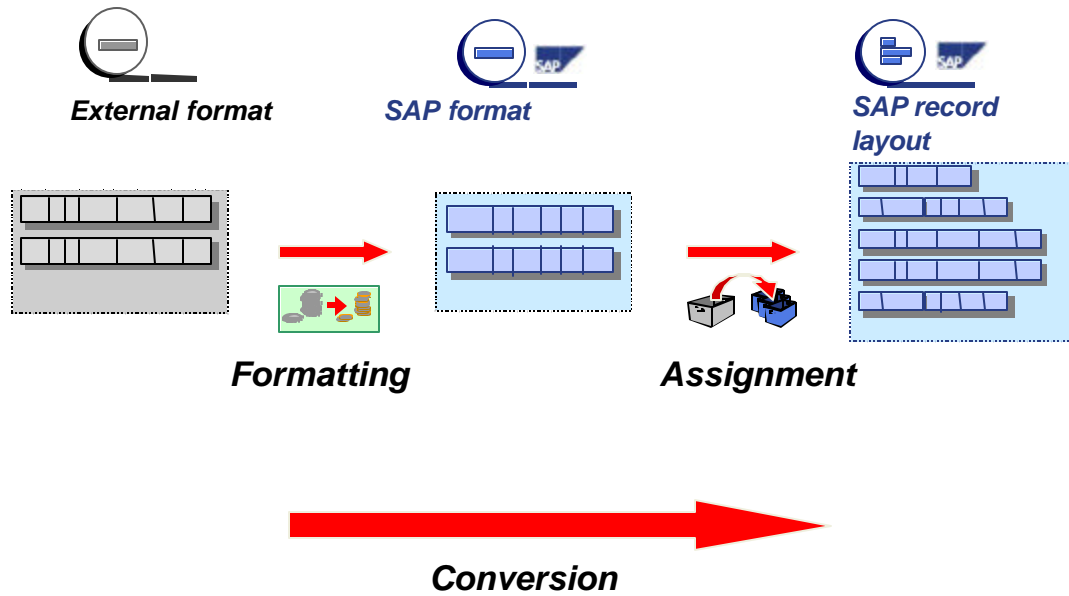
Application help

- General help on data transfer
- Data transfer techniques
- Data transfer before Release 4.6A
- Data transfer objects in applications

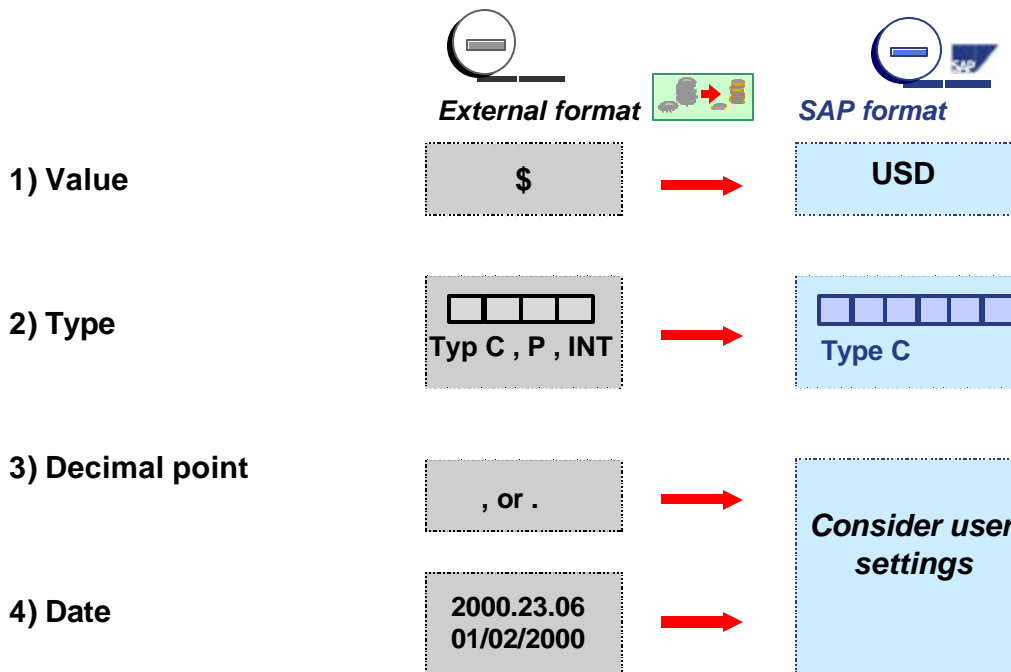


© SAP AG 1999

- The DX-WB documentation is in the DX-WB itself.



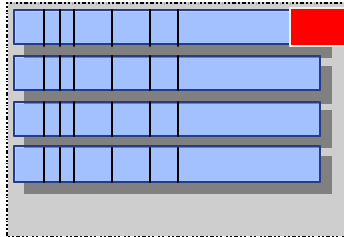
- The external data can be converted into the SAP record layout in one step or in two separate steps (formatting and assigning).
- There are important rules to follow for the formatting and assignment (next slides).



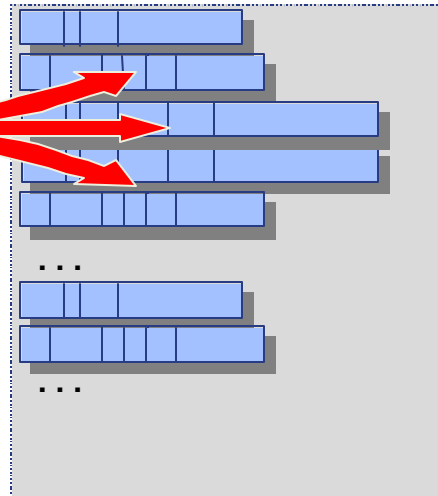
- If you use a transfer program, the external data must be formatted the same as for an online user who enters the values using the keyboard. The following rules apply:
 - The data must be in character format
 - The individual fields must not be longer than the defined SAP field.
 - If the data is shorter than the defined SAP field, you have to left align the data in the field (fill the right hand side with blank characters).
 - Note: If you use BAPIs and IDocs for the transfer, there might be exceptions.
- For conversions user-specific settings must be taken into account (e.g. fixed value settings in the user profile) except for BAPIs and IDocs.
User-specific settings are the date format and the decimal setting.
- If the data is transferred online, the authorizations and fixed values of the user who is starting the transfer are used.
- A user must be specified for background processing. This user should have the same user-specific settings. Occasionally this user might have different authorizations, for example, in the production system where the data is actually transferred.
- Note: Different authorizations and settings may result in errors when the data is transferred (for example, no authorization for the company code).



SAP Format



SAP Record Layout



How is the external data mapped to the record layouts?



- The standard transfer programs expects the file to be in the corresponding SAP record layout format.
- The data must correspond to this record layout structure. However, you do not have to use all of the record layout structures.
- Some structures must be used (required), while others are optional; this depends on the application. For information about which structures are required and which are optional, see the documentation for the record layout structures.

- The data records contained in the sequential file must be in a format that can be read by an SAP standard transfer method.
- Each transfer program expects the file to be in this general SAP record layout format.
- Each record layout contains individual structures, each of which is defined in the ABAP Dictionary.
- The structure descriptions and the layout of the sequential file depend on the application; the structures and layouts are described in the documentation for the standard transfer procedure.
- If a new session header is added to the structure, the existing session is closed and a new session is generated.

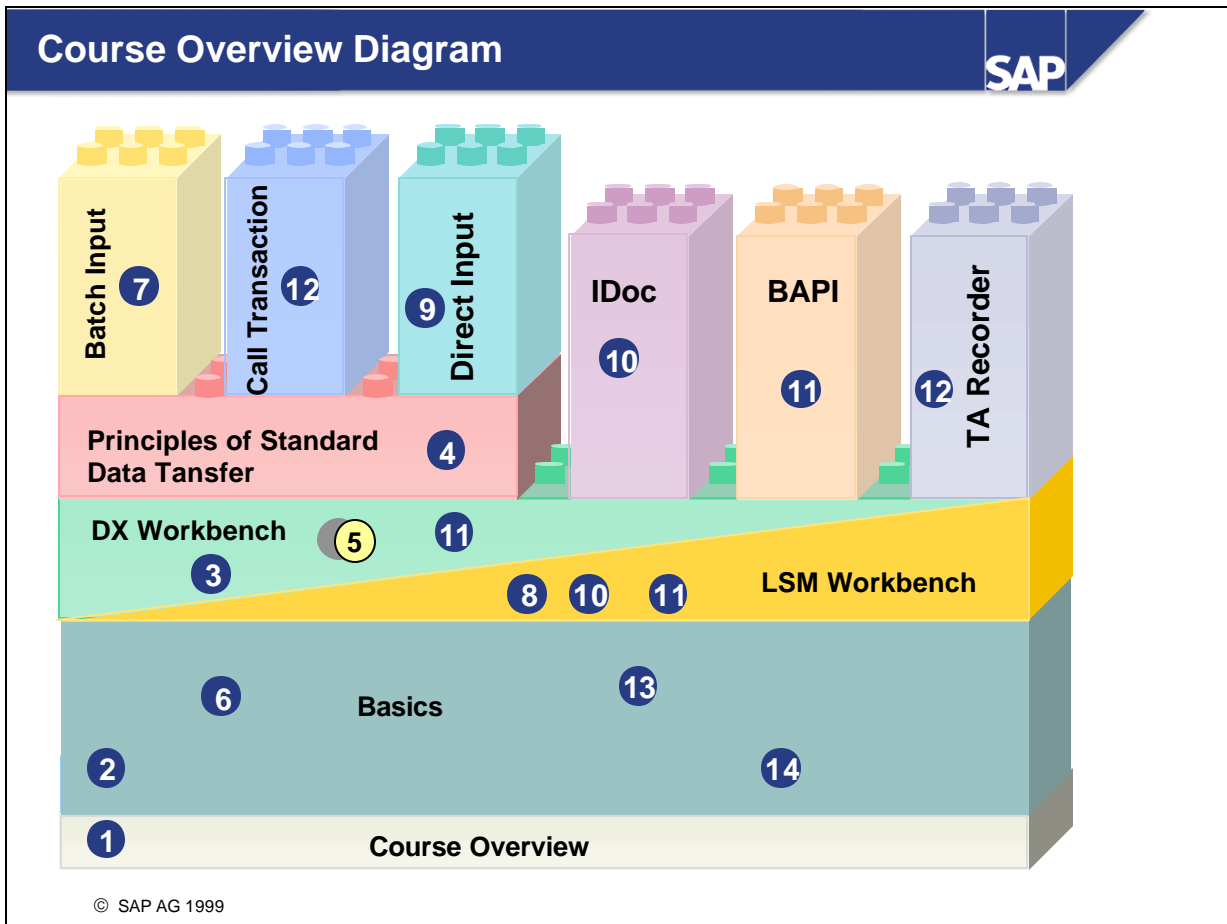
- Each session contains a session header (record type 0), also called a session header record. The session header record contains administration data about the batch input file to be created. All other records of the sequential file up to the next session header are assigned to the current session.
- Note: If the transfer is carried out in the background, the user specified in the session header for the authorization check is used. This means that the authorizations and settings of the fixed values in the user profile are used for checking.
- If the stop indicator is set (= 'X'), the processed session remains in the system, otherwise it is deleted.
- In the field NODATA you specify the character that will later be used as the NODATA indicator.

- Transaction header data
 - These structures hold data for **one transaction**. You enter the transaction and the entries you made on the initial screen of the transaction to this structure. This includes, among other entries, the transaction code and key fields such as account group, company code key, document type, account number, and document number.
- Transaction data
 - These structures contain the master record and document position data. This includes the addresses of your business partners and the document position amounts.
- The different structures are described using record types. For the transaction data, specify the structure name in addition to the record type.
- The structures for batch input are defined in the ABAP Dictionary, where you can display the individual structures.

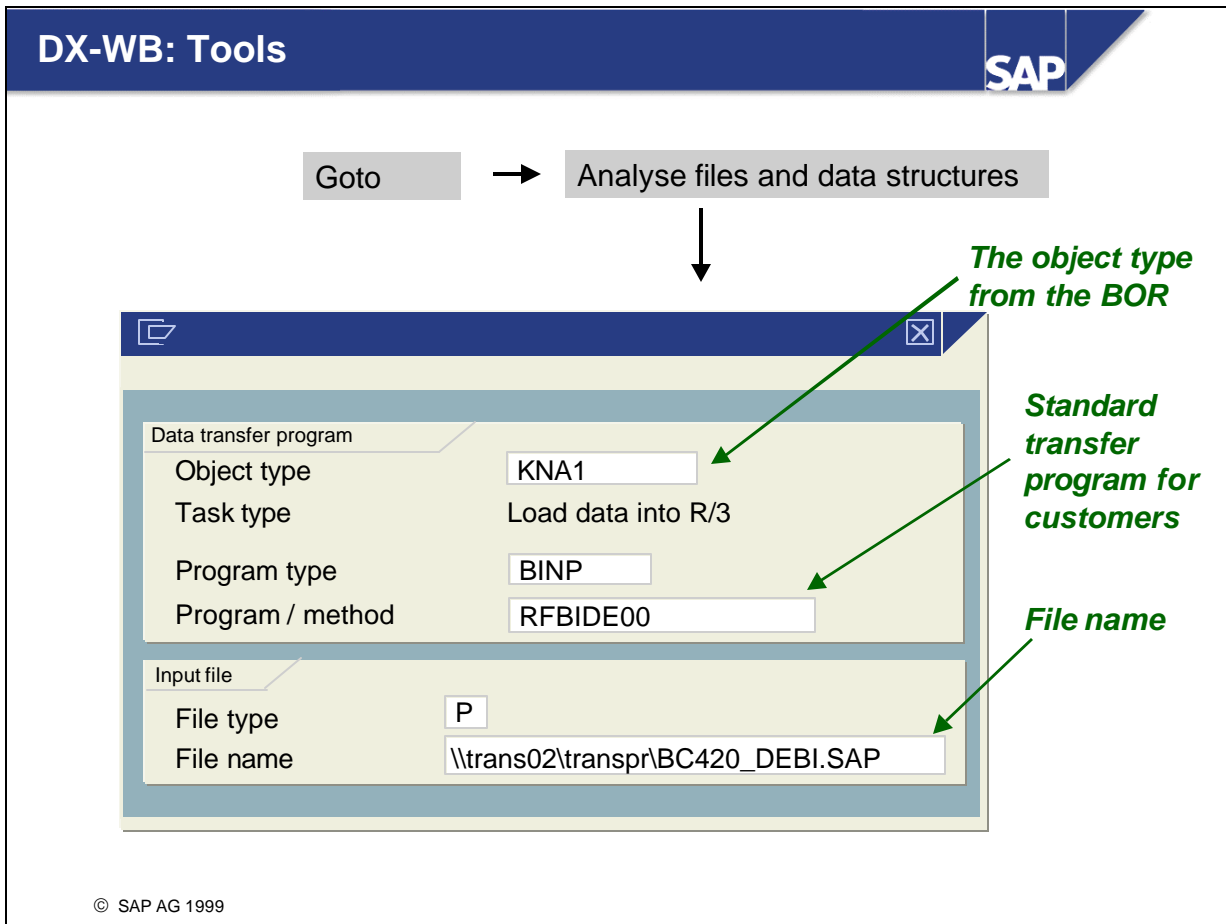
- Example of the record layout structures of the customer transfer.
- See the documentation for the description of the individual structures.

- Every application provides transfer programs. Which methods can be used (BI, CT, or DI) depends on the individual programs. You cannot influence this.
- If your data is in the SAP record layout structure and the relevant program supports several methods, you can use all of the methods without having to change the structure or the contents of the file.
- An example of this is program RFBIBL00, which transfers FI documents. You can use any of the three methods with this program.


- **Editing files using the DX-WB**
- **Creating test data for the data transfer**



- With some data transfer objects you can create a file with test data. Data from R/3 is inserted into a file.
- The function is available for standard transfer programs (BI,CT,DI) and BAPIs
- **Note: This function is not available for all data transfer programs. It is application-dependent.**
- See appendix 01 for an overview of the objects that can use this function.



- To use the tools of the DX-WB, you have to specify the program or the method for the data transfer.
- Proceed as follows:
 - Select the relevant object type in your application.
 - Select the supported program type.
 - Select the supported program or method.
 - Note: Use the F4 input help
- Then you must specify the file name and the directory in which this is stored or should be created. A physical or logical path/file name should be specified. (For more information about logical file names see the unit on sequential files)
- The path shown is valid for Release 4.6A
For the Release 4.6B the valid path is *Goto -> Analyze files and data structures.*

- Creates initial file under specified path/file name
- File contains all the structures of the object 
- Session header is filled

<i>Field Name</i>	<i>Initial Value</i>
STYPE	0
GROUP	<i>Object name</i>
MANDT	<i>Current client</i>
USNAM	<i>Current user</i>
START	..
XKEEP	
NODATA	/

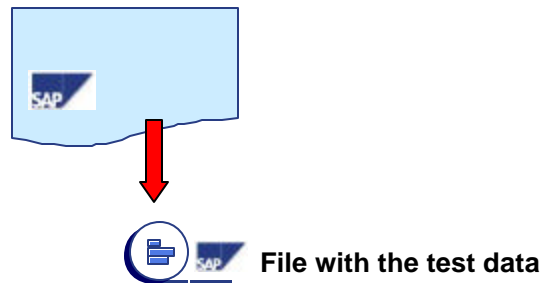
- Structures are preassigned with “/”.
Exception:

<i>Field Name</i>	<i>Initial Value</i>
STYPE	1 or 2
TBNAM	Structure name

© SAP AG 1999

- When you create a file for a selected object and method, this file is an initialized file. It contains all the structures that the object supports.
- If you are using a standard transfer program, the associated record layout always contains a session header.
- Structures are predefined with the NODATA character only for BI or DI. This is not required for BAPIs.

- Needs a program that can create test data
- The program must be provided by the application



(Overview: see appendix 01)

© SAP AG 1999

- With some data transfer objects you can create a file with test data. Data from R/3 is inserted into a file.
- **This function is not available for all transfer programs. It is application-dependent.**
- See appendix 01 for an overview of the objects that can use this function.
- The function is available for standard transfer programs (BI,CT,DI) and BAPIs.

Structure		Contents
BGR00	Session record	0Mappe1 003 ...
BKN00	Header date 1	
BKNA1	Standard	2BKNA1 / John
...		

Field Name	Short Description	Field Contents
STYPE	Record type	2
TBNAM	Table name	BKNA1
...		
Name1	Name 1	Smith
Name2	Name 2	/
ORT01	Ort	Berlin
LAND1	Land	DE
...	...	/

Structure editor enables the individual structures to be changed

Display/change

© SAP AG 1999

- The file display template contains the following information:
 - The number of data records is displayed in the title. This shows how many transactions the transfer file contains.
 - The fields of the transfer object are displayed in predefined length and sequence for each structure. So you can identify which fields of a structure are required in which length and sequence. By double clicking on a structure line you can see the fields of the structure.
 - The different hierarchy levels of a data record are indicated in different colors:
 - Hierarchy level 1:green
 - Hierarchy level 2:yellow
 - Hierarchy level 3: blue
 - Hierarchy level 4:gray
 - You can print the transfer file (*Table -> Print*)
- Note that only the first 100 characters of each structure are displayed in the overview.
- Note: This functionality applies only for BI and DI.

Function: Copy File **SAP**

Source

Application server

Remote server

File type

File name

Presentation server

File name

Target

Application server

Remote server

File type

File name

Presentation server

File name

All the structures of the selected transfer object can be copied

© SAP AG 1999

- The data transfer file can be copied between the application and presentation servers.
- The data transfer programs expect the transfer file to have a fixed length.
The following conversions are possible:
 - Copy without converting
 - The data transfer file is copied unmodified
 - Conversion of fixed length -> Delimited tabulator
 - This copy function can only be used to copy from the application server to the presentation server. The data is copied into a format and the data is delimited by a tabulator.
 - Conversion of delimited tabulator -> Fixed length
 - This copy function can only be used to copy from the presentation server to the application server. The data is copied to a fixed length.



Duplicate structure rows



Delete structure rows



Edit blocks in structure

Structure		Contents
BGR00	Session header record	0Session1 003 ...
BKN00	Header data 1	
BKNA1	General	2BKNA1 / John
...		

© SAP AG 1999

- You can edit the structure of an imported file as follows:
- Delete:
 - Place the cursor on the row you want to delete and choose *Delete row*.
- Copy:
 - Place the cursor on the row you want to duplicate and choose *Duplicate row*. The system duplicates the row and adds it to the structure.
- Select block:
 - Place the cursor on the first row of the block you want to edit.
 - Choose *Select block*. The block is selected and now has a gray background.
 - Place the cursor on the last row of the block you want to edit.
 - Choose *Copy* or *Cut*.
 - Place the cursor on the row in front of the location where you want to add the copied or cut block to.
 - Choose *Insert clipboard*.

The screenshot displays two windows from the SAP DX-WB interface. The left window shows a table with the following structure:

Structure		Contents
BGR00	Session record	0session1 003 ...
BKN00	Header data 1	
BKNA1	Standard	2BKNA1 / John
...		

The right window shows the 'Data record' form for table BKNA1. The record name is 'BKNA1 Customers'. The form contains the following fields:

Field name	Short Description	Field Contents
STYPE	Record type	2
TBNAM	Table name	BKNA1
...		
Name1	Name 1	Smith
Name2	Name 2	/
ORT01	Ort	Berlin
LAND1	Land	DE
...	...	/

A yellow arrow points from the 'BKNA1 Standard' row in the structure table to the 'Data record' form. A yellow callout box with the text 'Search help' is positioned over the 'Name 1' field.

Test data can be inserted into the record layout structure

- You can enter values for the individual fields in change mode. Most of the fields have input help.
- **Note:** All fields that are not needed must be filled with the NODATA character.

- Before you create test data the fields in the non-SAP system should be formatted and assigned to the fields in the SAP system. This is best done in a table.

- Fields in the SAP record layout structure that are not needed for data transfer, must be filled with the NODATA indicator. This means that these fields or field values will not be used during data transfer.
- If a field is not filled with the NODATA indicator, it will be transferred with the value SPACE. This could create severe problems, as the SPACE value may overwrite existing values in the system.
- Note: This only applies to batch input (BI) and direct input (DI).

- You can check the structure of a file.
 - If you have selected an object that uses the SAP standard transfer program, it is checked against the record layout structure.
 - If the file is in IDoc format, all the EDI inbound processing checks are carried out.
 - If a BAPI is used for the transfer, so that non-processed objects can be stored in an IDoc in R/3, your file must be free of errors. If you save objects with errors to a file, you can ignore all the messages about missing inbound partner profiles.

- After you choose *Structure @ Display*, a list of information about all included structures and fields appears. You can download this list to your PC.
- Some external mapping tools use this list for defining and generating the SAP target structures during conversion.

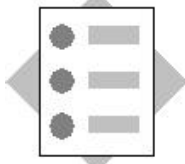
- After you choose *Structure* ® *Generate* you can translate SAP Structures (stored in the ABAP Dictionary) into other programming languages, such as Cobol, and store them in the file system as *Include files*.
- For this process, the SAP System supports the creation of transfer programs in C, COBOL, PL/1, and P_RPG. You use the ABAP program RDDSRG0 to generate the structure descriptions in the relevant programming language.

- The displayed utility considerably simplifies conversion in the external system.
- You can select the data transfer object for extracting the structure definitions. This sets the required SAP structures and creates them in a file in the programming language.

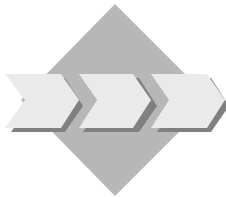


Unit: DX-WB Tools

Topic: Creating test data



- Creating test data in the DX-WB.



The address data for debtors have to be transferred. To do this, use program RFBIDE00. You have determined the fields or data to be transferred, in the external system. Identify the appropriate fields in the record layout structure.



File: BC420_DEB_##.TES

Subproject: DEB_##

Run: CT_TEST_##

Load Program: SAPBC420_PRPD_RFBIDE00_CT

- 1 First do the mapping (assignment of the fields) on a piece of paper. The external data should be mapped onto the record layout structure of program RFBIDE00. Afterwards a test data record must be generated using this mapping plan in the DX-WB. Create the run CT-TEST-## in the DX-WB and test the data record.

1-1 The database from the legacy system contains the following fields:

<u>Field name</u>	<u>Decription:</u>	<u>Length:</u>	<u>Type:</u>	<u>Field value:</u>
C_NO	Customer number	4	Char	4537
NAME	Customer name	30	Char	George Miller
SEARCH	Search term	20	Char	Miller
C_TYPE	Customer type	4	Char	0004
A_DATA	Street	40	Char	2907 Eagle Drive.
B_DATA	City	30	Char	Los Angeles
ZIP	Zip code	10	Char	92660
CO	Country	2	Char	US
SPO	Language	1	Char	E
TELF	Phone no.	16	Char	925-644-944
VAT	Sales tax identification number	20	Char	0302052345

Note: Use the necessary VAT no. for your country.

- 1-2 Create the **mapping plan** for the record layout structure of program RFBIDE00. The following list contains the necessary record layout structures and the most important fields :

BGR00, BKN00, BKNA1, BKNB1

- 1-3 Assign the fields of the legacy system to the fields of the SAP record layout structure (**mapping**). For assistance use the online transaction and read the field descriptions of the fields. Use the following abbreviations for the conversion method:

- F: Fixed value
- V: Variable
- M: Move
- M+C: Move + Convert

- 1-4 Under Coding/Comment fill in the necessary field value or the Rule/Comment.



For the assignment (mapping) of the data, the following is applicable for the fields of the record layout structure:

1. The name of the session = Debitor-##
2. Client = Logon client
3. User name = BC420-##
4. Customer number = Z-##-2200
5. Company Code = 0001
6. Account group = KUNA
7. Transaction code = XD01
8. Country = convert from U to US
9. Reconciliation account = 120000
10. Create old debtor number in the field BKNB1-ALTKN
11. The field customer type in the legacy data is not needed.
12. All remaining fields from 1-1

- 2 Start the DX-WB and go to the DX-WB tools.

- 2-1 Create the empty file BC420_DEB_##.TES

Choose the following settings:

- Object type: KNA1

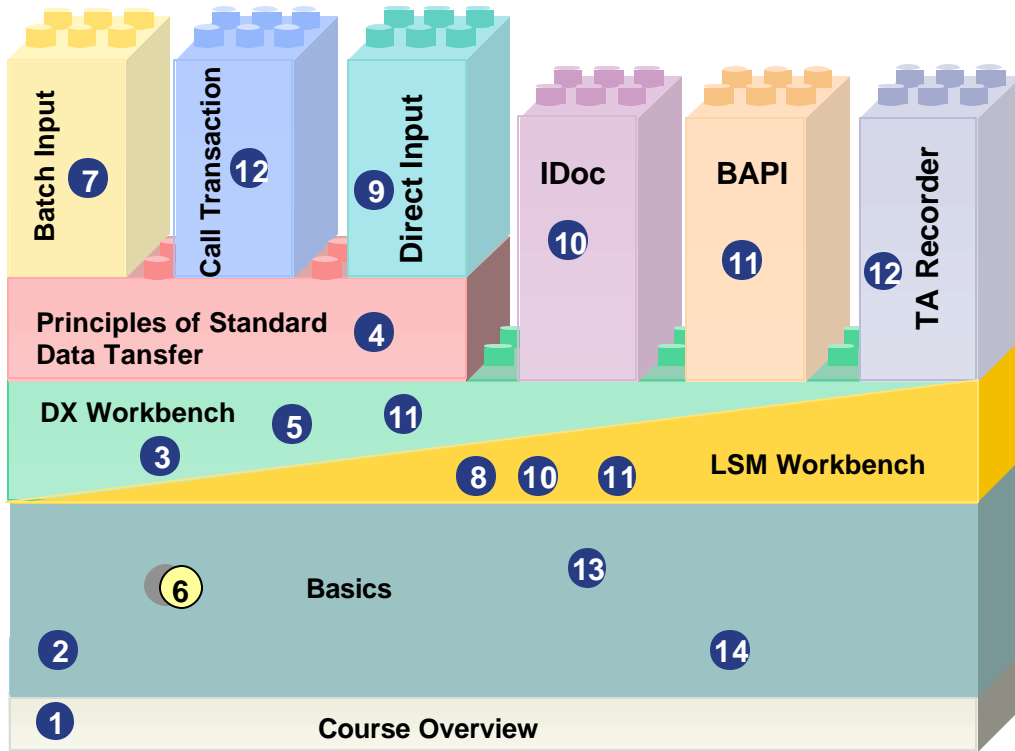
- Program type: BINP
- Program/Method: RFBIDE00

Note: Make sure you select the program type BINP and not REPO!

- 2-2 Delete the structures that are not needed (only the first four structures are needed!)
 - 2-3 Go to the structure editor. Copy the values from the list for a test data record.
 - 2-4 Save the file before leaving the editor.
 - 2-5 Check the structure (function: Check file)
-
- 3 Test the file in a run.
 - 3-1 Create the run CT_TEST-## with the description *Test with CT* under your project and the subproject DEB-##.
 - 3-2 In the run CT_TEST-##, create the following task:
 - Task: IMP_TEST-##
 - Description: *Test Transfer*
 - Task type: LOA
 - Program type: REPO
 - 3-3 Select the program SAPBC420_PRPD_RFBIDE00_CT.
Use the file name you created before: BC420_DEB_##.TES
 - 3-4 Start the run and confirm the tasks manually.
 - 3-5 Take a look at the log!
-
- 4 Use the transaction FD03 (Display debtor) to check if the debtor has been created correctly.

- **File Monitor**
- **Sequential Files**
- **Local Sequential Files**
- **Logical File Names**

Course Overview Diagram



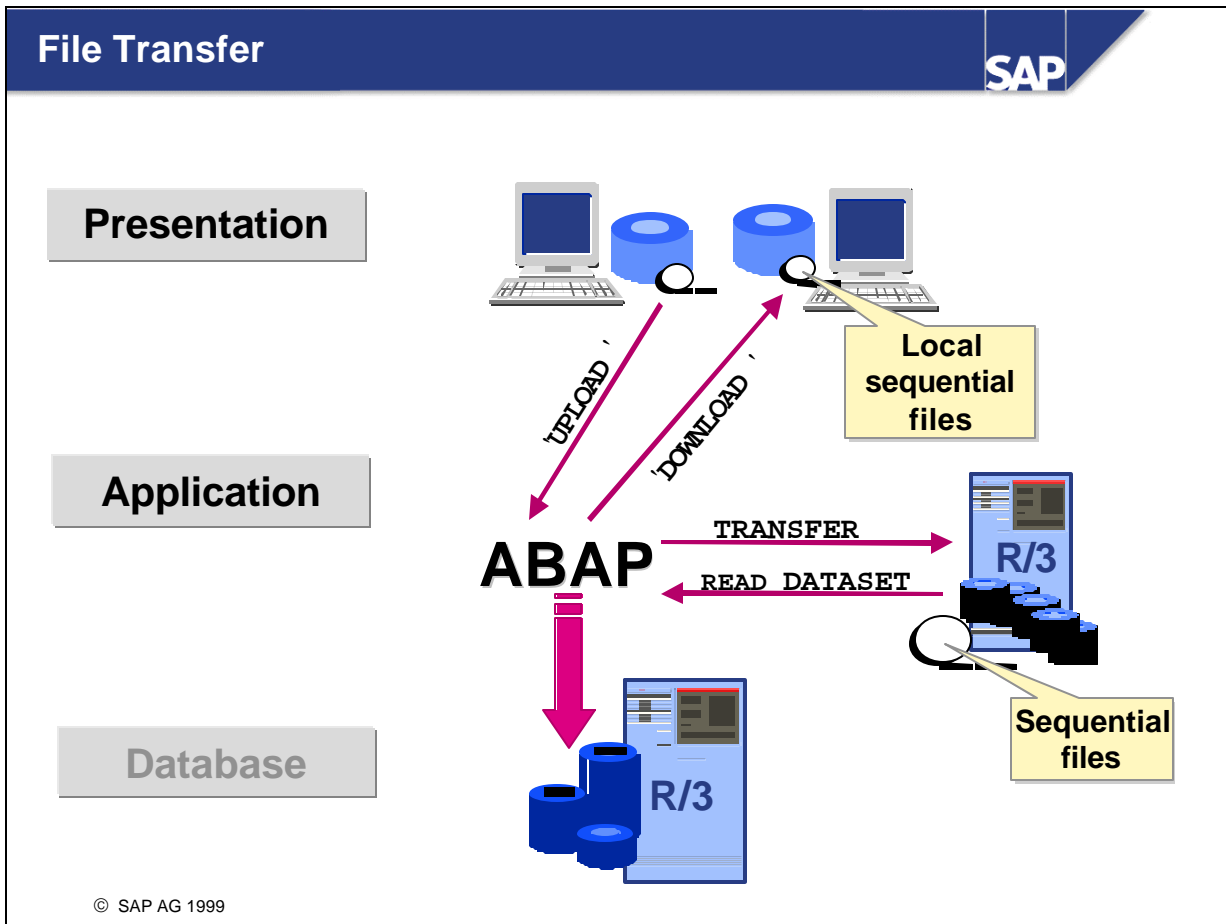


Overview & File Monitor

Sequential Files

Local Sequential Files

Logical File Names

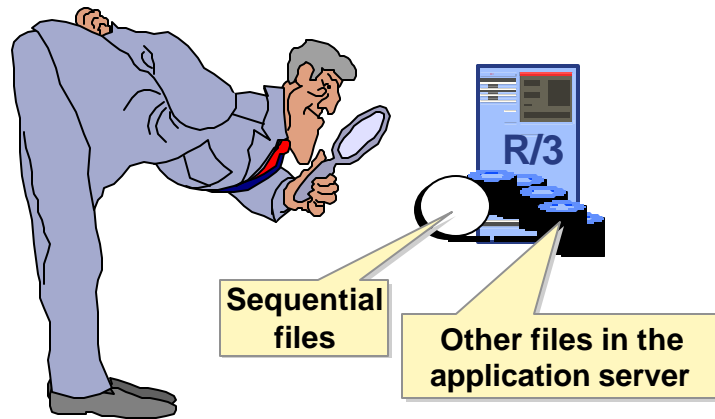


- The runtime environment, implemented on the application server, executes the ABAP programs. ABAP supports file transfer on the application server and on the frontend computers.
- The processing of the sequential files on the **application server** is supported by using ABAP language elements.
- You process sequential files with the ABAP commands **OPEN DATASET**, **READ DATASET** (read records), **TRANSFER** (write records), and **CLOSE DATASET**.
- The file interface on the **presentation server** is implemented by function modules.

The data is transferred from the application server to the database server with batch input (BI), Call Transaction (CT) or Direct Input (DI).

- Enables you to view directories and files in the application server
- The data in the application server cannot be changed

Application



© SAP AG 1999

- The file monitor is a CCMS tool (Computing Center Management System). The administrator can use the CCMS to monitor and control the R/3 server.
- With the file monitor you can display the R/3 directory structure - the database directories and files are excluded here. You could, for example, view the cross-application server transport directory "\trans", or the instance-specific directories "\work", "\data", "\DVEBMGS...".

Directories in the File Monitor

SAP

27.07.1999 15:57:58 I41 iwdf4041

SAP Directories

...

DIR_EXECUTABLE	D:\usr\sap\I41\SYS\exe\run
DIR_GLOBAL	D:\usr\sap\I41\SYS\global
DIR_GRAPH_EXE	D:\usr\sap\I41\SYS\exe\run
DIR_GRAPH_LIB	D:\usr\sap\I41\SYS\exe\run
DIR_HOME	D:\usr\sap\I41\DVEBMGS00\work
DIR_ORAHOME	unknown
DIR_PAGING	D:\usr\sap\I41\DVEBMGS00\data
DIR_PROFILE	D:\usr\sap\I41\SYS\profile
DIR_PROTOKOLLS	D:\usr\sap\I41\DVEBMGS00\log
DIR_SOURCE	D:\usr\sap\I41\SYS\src
DIR_TEMP	.
DIR_TRANS	\\iwdf4041\sapmnt\trans
...	

*DX-WB and LSMW
working directory*

*Global R/3 directory parameters and their
physical directories*

*When double-click
directory, see next
slide ...*

© SAP AG 1999

- There are two columns on the initial screen of the file monitor:
 - On the left, the system-internal directory parameters
 - On the right, the physical directories of these parameters.
- "DIR_TRANS" represents the global transport directory used for transporting programs and other objects between systems in the R/3 System landscape.
- To go to the directories displayed here, double-click the directory or choose *Display*.
- Note: the data files for the exercises in this course were placed in the home directory.

Directory \\iwd4041\sapmnt\trans

Name	Length	Creator	Last Change
...			
BC420TEST	690	Administ	17.07.1999 10:54
BDETEST	324590	Administ	17.07.1999 10:54
TESTFILE	1120	Administ	17.07.1999 10:54
bin	0	Administ	11.01.1999 09:11
buffer	0	Administ	11.01.1999 09:11
cofiles	0	Administ	11.01.1999 09:11
data	0	Administ	11.01.1999 09:11
eps	0	Administ	11.01.1999 09:11
etc	0	Administ	11.01.1999 09:11
log	0	Administ	11.01.1999 09:11
.....			

*Double-click on file
see next file ...*

Files in this directory

Other sub-directories of the transport directory

- You can find **files** and other **subdirectories** in the selected transport directory. (bin, buffer, cofiles, etc.); double-click again to go to these subdirectories. Subdirectories have the length specification "0").
- At this level the contents of selected files can be displayed via "select" or double click.

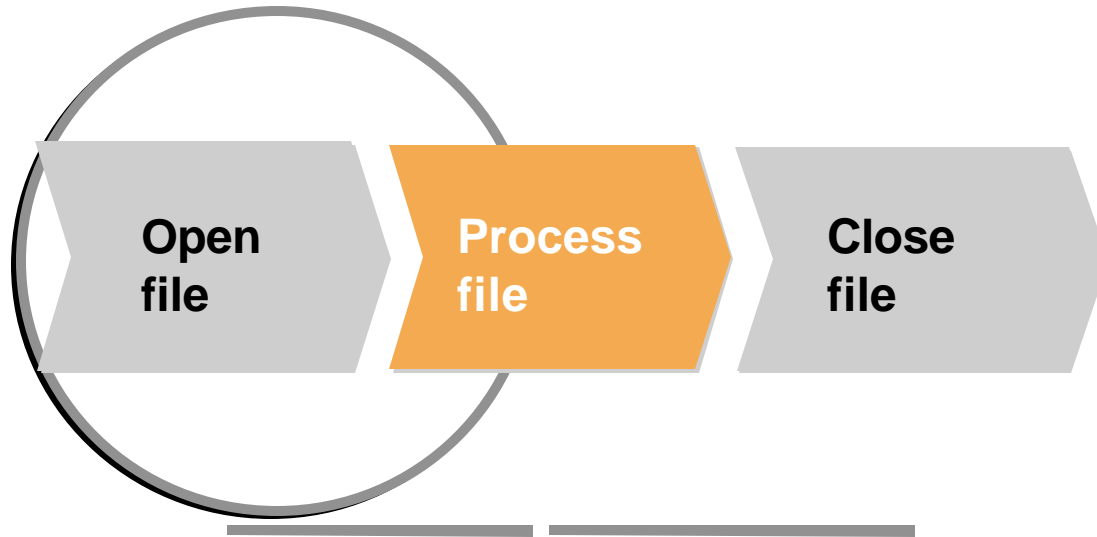
Overview & File Monitor



Sequential Files

Local Sequential Files

Logical File Names



© SAP AG 1999

- Before data records can be written to a sequential file or read from a file, the file has to be opened.
- After processing the file is closed again.

- A file processing program must define the required structures for the data records using a TABLES or DATA statement. These structures are used as program-internal work areas for the data records.
- The OPEN statement is used to begin file processing and **open** the source and target sequential files for reading or writing.
- **Reading:** The READ DATASET statement is used to **read** the records from the source file into the data structures for later processing in the program.
- **Writing:** TRANSFER statements are used to **transfer** the filled structures to the target file.
- The CLOSE DATASET statement is used to end file processing and **close** the sequential files.

- See slide “Read file“.

- You can explicitly open a file using the command OPEN DATASET <file name>.
- If the file can be opened, SY-SUBRC is set to 0, otherwise it is set to 8. Any errors are ignored.
- We recommend, for reasons of clarity, that you open the file explicitly. Using the OPEN DATASET statement allows you to test whether an error has occurred when you open the file (SY-SUBRC = 8). If you do not explicitly open the file, a READ DATASET or TRANSFER command will attempt to open the file using the default mode. In this case, if an open error occurs, you cannot trap the error and the program will terminate processing.
- Possible additions:
 - FOR INPUT
Opens an existing file to read it. If the file was already open, the file is read from the beginning. FOR INPUT is the default.
 - FOR OUTPUT
Opens the file to write it. If the file exists already, the contents are deleted. If the file does not exist, it is created.
 - FOR APPENDING
Opens the file to append records to the end of the file. If the file does not exist, it is created. If the file exists and was already open, the pointer to the next record is moved to the end of the file.

- When you open a file, you can choose between BINARY and TEXT mode.
 - IN BINARY MODE - the length of the data input from a READ DATASET or output from a TRANSFER is dependent on the length of the structure being used in the operation.
 - IN TEXT MODE - the length of the data input from a READ DATASET or output is dependent on the end of record indicator.
 - Without an addition the file in the binary mode is opened.
- Th OPEN command can also return a message by using the addition MESSAGE <field>. The associated operating system message is stored in the field <field>, if there is an error opening the file.

- The above diagram helps explain the difference between binary mode and text mode. In each example three records of different lengths are transferred. Then the data is imported into three structures of the same length.
 - In the text mode an end of record indicator specific to the operating system is set after each data record is written. When reading the file, the end of record indicator is used to determine the length of data to be read.
 - In binary mode data is transferred according to the length of the source or target structure.
- Note: Blank characters at the end of a data record are not deleted in text mode.

- The TRANSFER command is used to write a data record into a sequential file.
- A field or structure would be filled with data before each TRANSFER command.
- <field> can be a field or a structure.
- The execution of the TRANSFER command depends on the mode: -
 - - Binary mode: Appends the length of the field or the structure to the file
 - - Text mode: Writes a new record to the file
- If the specified file is not opened, TRANSFER attempts to open the file FOR OUTPUT (IN BINARY MODE or with the further specifications of the last OPEN command for this file). If this does not work, a runtime error results.
- Any errors will result in the program terminating.
- The additional parameter LENGTH <len> permits an explicit length specification in bytes. In this case the exact number of characters specified in <len> are transferred. If the structure is shorter, it is padded (in text mode with blank characters and in binary mode with hexadecimal zeros). If the structure is longer, it is truncated.

- With READ DATASET you can read a record from a sequential file into a field or structure.
- Possible structures are field strings or table work areas.
- The execution of the READ DATASET command depends on the mode: -
 - - Binary mode: Reads the structure length
 - - Text mode: Reads one record of data
- If the specified file is not opened, READ DATASET attempts to open the file (IN BINARY MODE FOR INPUT) or with the additions of the last OPEN DATASET command for this file).
- If the file end has been reached, SY-SUBRC is set to 4, otherwise it is set to 0. If the file cannot be opened, SY-SUBRC is set to 8. Any errors will result in the program terminating.
- READ DATASET, like TRANSFER, does not execute any implicit conversions. The data is imported as it has been written.
- The READ DATASET command together with the additional parameter LENGTH <len> returns the length of the imported file record into the field <len>.

- The command CLOSE DATASET <file name > closes a sequential file. Errors are ignored the same as with OPEN DATASET.
- Within a program when the screen is changed all open files are shut and opened again with their old status when processing is started again.
- All files are closed when the program is exited.
- We recommend, for reasons of clarity, that you open and close the file directly. OPEN DATASET has also the advantage that if an error does occur when you open the file (SY-SUBRC = 8), it will not result in a termination.
- A file is not implicitly closed, if READ DATASET reaches the end of the file.
- With the command DELETE DATASET <file name> you can physically delete a sequential file. The file will then not longer exist in the operating system. Once it has been deleted, SY-SUBRC is set to 0.
- The current file status can be displayed in debugging.

- SAP record layouts must be filled with the NODATA indicator, before they are described with data.
- If fields in the SAP System are not to be filled with external data, the standard transfer process expects this special character (NODATA indicator) for record layout fields. The default special character is '/', but it can be redefined in the field BGR00-NODATA. You have to initialize the record layouts with this special character.

- As described on the slide, conversion of external data to the SAP record layout format, which strictly requires the use of the standard transfer method, includes several steps:

1. Open all files.
2. Initialize help structures with the NODATA character. These structures are copied to the record layouts to initialize the record layouts.
3. Read an external data record.
4. Initialize the record layouts with the previously initialized help structures.
5. Fill the record layouts with the external data:
First the external data is formatted (for example "\$" → "USD"). The formatted data is then mapped to the corresponding fields of the record layout.
6. Write the filled record structures to the target file.
7. Loop back to step 3 until end of file.
8. Close the files.

■

- Initialization of the record layout with the NODATA character is always executed using the same procedure.
- All structure fields are addressed and initialized using the pointer operations "ASSIGN COMPONENT ... TO <f>. MOVE ... TO <f>".
- The form routine INIT is designed so that all structures can be transferred to the formal interface parameter P_REC; all structures will then be initialized with the pre-defined NODATA character.

- The code shows the important steps from initializing the help structures through to writing the record layouts.

- Example of formatting is converting the language REC_LEGACY-SPRAS from the old format (e.g. "E") into the new SAP format ("EN").
- The formatted language is assigned to the record layout REC_CONVERT-SPRAS.

- Using the function module DOWNLOAD you can transfer the contents of an internal table to a local file.
- Using the function module UPLOAD you can transfer the contents from a local sequential file into an internal table.
- Enter the whole file path and name of the local file (e.g. '/tmp/myfile' for a Unix file and e.g. 'C:\MYFILE.TXT' for a PC file).
 - The presentation server must know the directory.
 - Customers can choose an appropriate file name.
- The system asks you for the file name and file type.
- The data is then converted into the appropriate file type.

- For the function module `DOWNLOAD` you need an internal table for the data transfer. You can define this table to fit your data structure and then fill it with your data.
- With the parameter `MODE` you can decide which write mode is used ('A' for extend file, ' ' for create new file).
- If you want, you can specify default values for the file name and file type and enter a title for the file dialog.
- The `IMPORT` parameters specify the actual values entered by the user for the file name and file type and also the number of bytes transferred.
- If the function module `DOWNLOAD` stores a file under DOS, the export parameter `CODEPAGE` must be set to 'IBM'. The parameter has no other meaning.
- All `EXPORTING` parameters are optional.
- If a binary file is to be created, the file length must be specified. In this case the transfer table must consist of one column of type X (hexadecimal numbers).

- In the coding example the `SELECT` commands reads customer data from table `KNA1` and puts it into an internal table that has already been defined.
- Then the function module `DOWNLOAD` stores the internal table on the presentation server.

- For the function module `UPLOAD` you need an internal table for the data transfer. You can define this table to fit your data structure at the start of the program.
- If you want, you can specify default values for the file name and file type and enter a title for the file dialog.
- If the function module `UPLOAD` stores a file under DOS, the export parameter `CODEPAGE` must be set to `'IBM'`. The parameter has no other meaning.

- In the coding example the function module UPLOAD reads a local sequential file from the presentation server and puts it in an internal table that has already been defined.
- Then the internal table is output using LOOP.

- A common use of logical file names is when archiving of R/3 application data using the transaction "SARA". However, logical file names can also be useful for external data transfer programs.
- Using the platform-independent, logical file name you can specify the file name and menu path under which files are to be created. According to the operating system used, the physical paths and file names specified in Customizing (transaction "FILE") are used.
- The portability of programs can be implemented using a function module (FILE_GET_NAME).

- You can create logical paths and files using the Customizing transaction "FILE".
- Platform-specific physical path must contain the reserved word <FILENAME> as the placeholder for the file name. It may also contain other reserved words - see the documentation (F1 help).

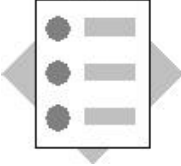
- You can use reserved words that are replaced at runtime by current values. These reserved words are always contained within angle brackets.
- You can see an overview of all the reserved words in the documentation about logical file names (F1 help).

- The file name form is dependent on the operating system. Portable programs can be implemented by using the function module **FILE_GET_NAME**. This function module provides the physical file name associated with a specified logical file name. You can find a description of the function module in the documentation for the Function Builder (Library of function modules).

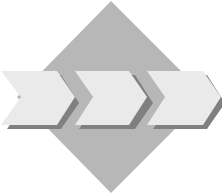


Unit: Sequential Files

Topic: Working with Sequential Files / Creating a Conversion Program for Debtor Data



- Writing of a program that reads a sequential file with legacy data, that converts the legacy data to the record layout format and then writes in an output file.
- The data transfer from the DX-WB must be started with this output file. The result is a BI session, which is run later.



The analysis of the transfer process to be selected, as well as the analysis of the SAP record layout structures has been completed. You should write a conversion program now, in order to convert the legacy data in the SAP record layout format.



Program: ZBC420_##_CONVERT_DEBITORS

Template program: SAPBC420_SEQT_CONVERT_DEBI1 oder
SAPBC420_SEQT_CONVERT_DEBI2

Solution: SAPBC420_SEQS_CONVERT_DEBI

If time is short point 2 is optional

- 1 Copy program `SAPBC420_SEQS_CONVERT_DEBI` to name `ZBC420_##_CONVERT_DEBI` and activate the program.
(by doing this point 2 does not apply)

- 2 **Optional:** convert the debtor legacy data.

Write a program `ZBC420_##_CONVERT_DEBITORS` for the import and conversion of debtor legacy data of the source file `BC420_DEBI.LEG`

The path is automatically generated in the template program. You do not need to set it separately. Write the file `BC420_##_DEBI.SAP` and use it as an output file.

The necessary source and target structures are already set in INCLUDE files, too. There are two template programs. You should use one of them as a template for your solution.

`SAPBC420_SEQT_CONVERT_DEBI1` contains a large number of notes concerning the statements that are still missing. Please complete them.

`SAPBC420_SEQT_CONVERT_DEBI2` contains fewer statements and therefore should be used by participants with a sound knowledge of ABAP.

- 2-1 The opening and closing of the sequential files is specified according to the template program (the data variables *infile*, *outfile* and *errfile* are already predefined with the matching path and can be used directly).
- 2-2 Form routines to initialize and fill the record layouts are already prepared and partly contain coding. Please complete the coding.
- 2-3 Read the legacy data in a loop. Build the record layout structures in this loop. Save the record layout structures in the target file. Incorrect records are records in which the old debtor number has not been filled in (filled with a blank character). Write these incorrect records in the error file.
 - 2-3-1 You have to format some fields before you assign them to the record layout fields:
 - 2-3-2 Assign the old debtor number to the field BKNB1. Create the debtor in the system with a new legacy number. The new number should be a reformatted 10-digit debtor number. Use "Z-##-1" as a prefix. Because of the predefined variable *auxkunnr(10)*, this has already been performed. Fill in the last four digits of this new debtor number using the loop index SY-INDEX (numbers 0001-0010 are supplied via a work field NUM). This is how the following debtor numbers are created: Z-##-10001 bis Z-##-10010.
 - 2-3-3 For another formatting example change the language key (SAP language is 2 digits). HINT: Use or write a form routine *CONVERT_LANGUAGE*, in which the language key is converted in a CASE statement. Add the following languages: D->DE, F->FR, S->ES, J->JA, E->EN.
- 2-4 When the program has been created, check the structure of your record layout file with the AL11 and the DX-WB.

3 Create a new run with two tasks under your subproject:

Project name: *use old project.*
 Subproject name: *use old subproject.*
 Run definition name: *BI-##, transfer debtors*

- 3-1 To be able to use your own programs in the DX-WB, you have to register them first. Register your program ZBC420_##_CONVERT_DEBITORS as task type MAP under the program type REPO.
- 3-2 Create the following task:
 Technical name : *map-##.*
 Description: Map debtors
 Task type: MAP
 Select your mapping program from (the program type REPO)!
- 3-4 Create the following task:
 Technical name : *Debi-##-load.*
 Description: Load debtors
 Task type: LOA
 Select the standard transfer program RFBIDE00 with program type BINP.

3-5 Start the run and analyze the flow trace.

Important: When you perform the task LOA, check the file first (flag checkbox). If there is no termination situation discovered, the session can be created.

Note: **Do not run the session yet !**



Unit: Sequential files

Solution SAPBC420_SEQS_CONVERT_DEBI

```
*&-----*
*& Report  SAPBC420_SEQS_CONVERT_DEBI          *
*&-----*
*& Source file contains debtors in legacy format. *
*& This data is converted into SAP record layout format for data *
*& transfer with RFBIDE00                       *
*&-----*
```

REPORT SAPBC420_SEQS_CONVERT_DEBI message-id bc420 LINE-SIZE 200.

TABLES: bgr00, bkn00, bkna1, bknb1.

* include contains structur of file with SAP-formatted data

INCLUDE sapbc420_seqI_debi_legacystruc.

INCLUDE bc420_fill_nodata.

* help structure (structure equals BKN00)

DATA bkn00_nodata LIKE bkn00.

* help structure (structure equals BKNA1)

DATA bkna1_nodata LIKE bkna1.

* help structure (structure equals BKNB1)

DATA bknb1_nodata LIKE bknb1.

DATA: num(4) TYPE n,

text(100).

PARAMETERS:

newkunnr(10) DEFAULT 'Z-##-1',

"will be new debtor-no.

infile(70) LOWER CASE,

outfile1(70) LOWER CASE,

errfile(70) LOWER CASE,

session(20) DEFAULT 'BC420SES-##'

LOWER CASE,

nodata DEFAULT '/'

LOWER CASE.

INITIALIZATION.

*-----

INCLUDE bc420x i nath include


```
CONCATENATE path 'BC420_DEBI.LEG' INTO infile.
CONCATENATE path 'BC420_##_DEBI.SAP' INTO outfile1.
CONCATENATE path 'BC420_##_DEBI.ERR' INTO errfile.
```

AT SELECTION-SCREEN.

*-----

* open files

```
OPEN DATASET: infile FOR INPUT IN TEXT MODE MESSAGE text.
```

```
IF sy-subrc NE 0.
```

```
    MESSAGE e101 WITH infile.
```

```
ENDIF.
```

```
OPEN DATASET: outfile1 FOR OUTPUT IN TEXT MODE MESSAGE text.
```

```
IF sy-subrc NE 0.
```

```
    MESSAGE e100 WITH outfile1.
```

```
ENDIF.
```

```
OPEN DATASET: errfile FOR OUTPUT IN TEXT MODE MESSAGE text.
```

```
IF sy-subrc NE 0.
```

```
    MESSAGE e100 WITH errfile.
```

```
ENDIF.
```

START-OF-SELECTION.

*-----

* fill structure with nodata

```
PERFORM init USING bkn00_nodata nodata.
```

```
PERFORM init USING bkna1_nodata nodata.
```

```
PERFORM init USING bknb1_nodata nodata.
```

* fill structure bgr00

```
PERFORM fill_bgr00.
```

* write data to file

```
TRANSFER bgr00 TO outfile1.
```

```
WRITE / bgr00.
```

* fill structure bkn00, bkna1 and bknb1

```
DO.
```

```
    READ DATASET infile INTO rec_legacy.
```

```
    IF sy-subrc NE 0. EXIT. ENDIF.
```

```
    " just to do some cleansing of legacy data!
```

```
    IF rec_legacy-kunnr is initial.
```

```
        transfer rec_legacy to errfile.
```

```
        WRITE: / text-001, rec_legacy-kunnr.
```

```
    ELSE.                                " record is okay !
```

```
ENDDO.
```

```

PERFORM fill_bkn00.
MOVE  bkna1_nodata TO bkna1.
PERFORM fill_bkna1.
MOVE  bknb1_nodata TO bknb1.
PERFORM fill_bknb1.
" write data to file
TRANSFER: bkn00 TO outfile1,
          bkna1 TO outfile1,
          bknb1 TO outfile1.
WRITE: / text-002, rec_legacy-kunnr.
WRITE:  text-003, bkn00-kunnr.
endif.
ENDDO.
* close files
CLOSE DATASET: infile,  outfile1,  errfile.

*&-----*
*&      Form  FILL_BGR00
*      fill structure bgr00
*-----*
FORM fill_bgr00.
MOVE: ' 0'          TO bgr00-stype,
      session      TO bgr00-group,
      sy-mandt     TO bgr00-mandt,
      sy-uname     TO bgr00-usnam,
      nodata       TO bgr00-nodata,
      ' X'         TO bgr00-xkeep.
ENDFORM                                " FILL_BGR00
*&-----*
*&      Form  FILL_BKN00
*&      fill structure bkn00
*&      New debtor-numbers are created; they start with "Z-##- 1"
*&      the old debtor number is concatenated --> "Z-##- 1xxxx"
*-----*
FORM fill_bkn00.
MOVE: ' 1'          TO bkn00-stype,
      ' XD01'       TO bkn00-tcode,
      ' 0001'       TO bkn00-bukrs,
      ' KUNA'       TO bkn00-ktokd.
num = sy-index.      "sy-index is set in D0-ENDDO-Loop
MOVE num TO newkunnr+6(4).  "move new number

```

```

ENDFORM                                " FILL_BKN00
*&-----*
*&      Form FILL_BKNA1
*      fill structure bkna1
*-----*

FORM fill_bkna1.
  MOVE: ' 2'                TO bkna1-stype,
        ' BKNA1'          TO bkna1-tbnam,
        rec_legacy-name1  TO bkna1-name1,
        rec_legacy-sort1  TO bkna1-sort1,
        rec_legacy-stras  TO bkna1-stras,
        rec_legacy-ort01  TO bkna1-ort01,
        rec_legacy-pstlz  TO bkna1-pstlz,
        rec_legacy-land1  TO bkna1-land1,
        rec_legacy-telf1  TO bkna1-telf1.

        perform convert_language. "D->DE; E->EN and so on

  IF rec_legacy-stceg IS INITIAL.
    MOVE nodata          TO bkna1-stceg.
  ELSE.
    MOVE rec_legacy-stceg TO bkna1-stceg.
  ENDIF.

ENDFORM                                " FILL_BKNA1
*&-----*
*&      Form FILL_BKNB1
*      fill structure bknb1
*-----*

FORM fill_bknb1.
  MOVE: ' 2'                TO bknb1-stype,
        ' BKNB1'          TO bknb1-tbnam,
        ' 120000'        TO bknb1-akont,
        rec_legacy-kunnr  TO bknb1-altkn.
*      write: / 'Old number = ', rec_legacy-kunnr.

ENDFORM                                " FILL_BKNB1

*-----*
*      FORM Convert_langu
*-----*

FORM convert_language.
  CASE rec_legacy-spras.

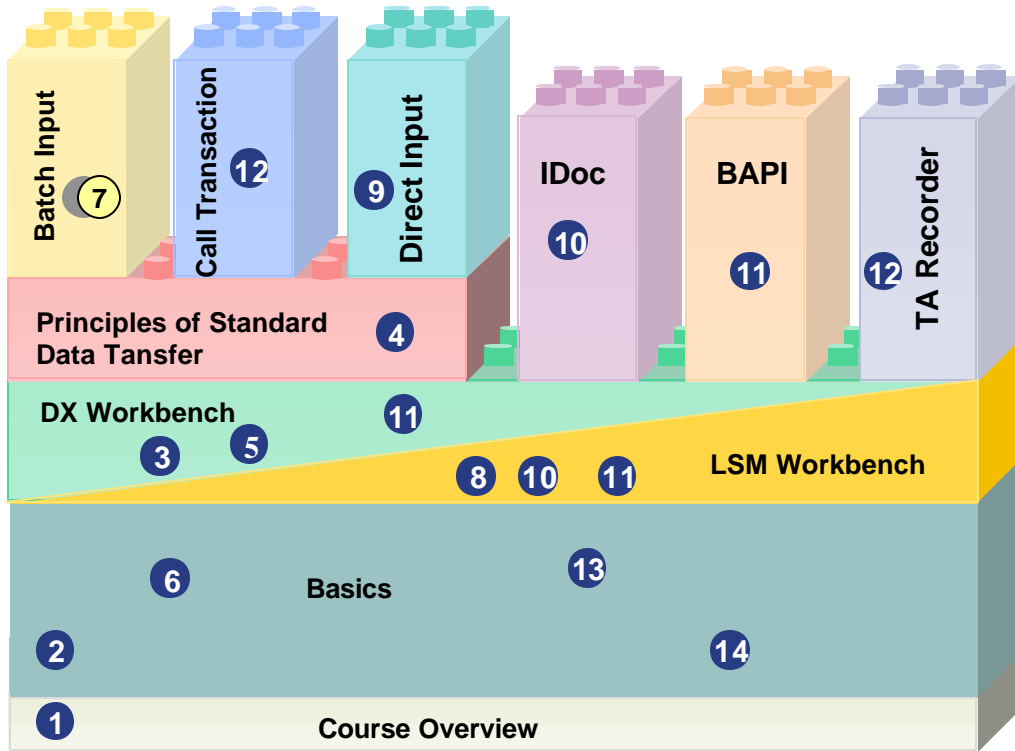
```

```
      bkna1-spras = ' EN' .  
WHEN ' D' .  
      bkna1-spras = ' DE' .  
WHEN ' F' .  
      bkna1-spras = ' FR' .  
WHEN ' S' .  
      bkna1-spras = ' ES' .  
WHEN ' J' .  
      bkna1-spras = ' JA' .  
ENDCASE.  
ENDFORM
```

Contents:

- **Batch input sessions**
- **Batch input monitor**

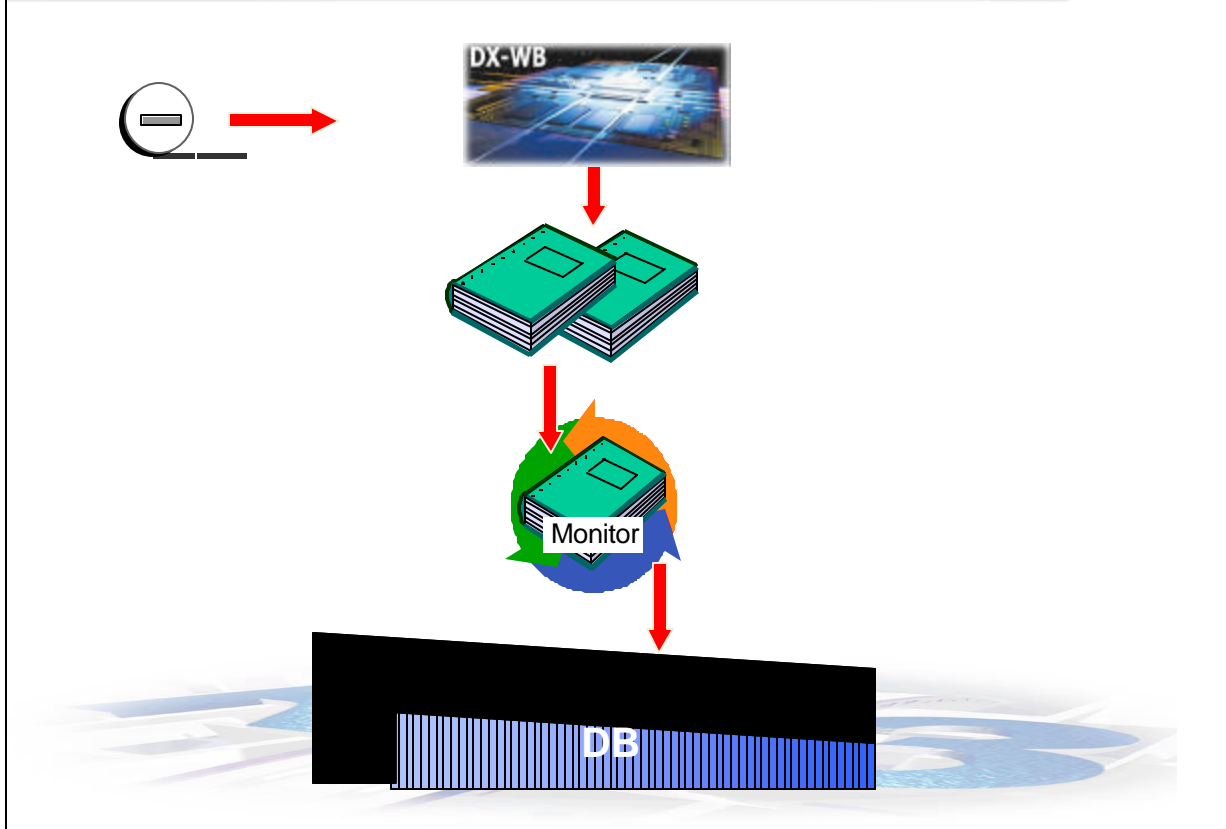
Course Overview Diagram



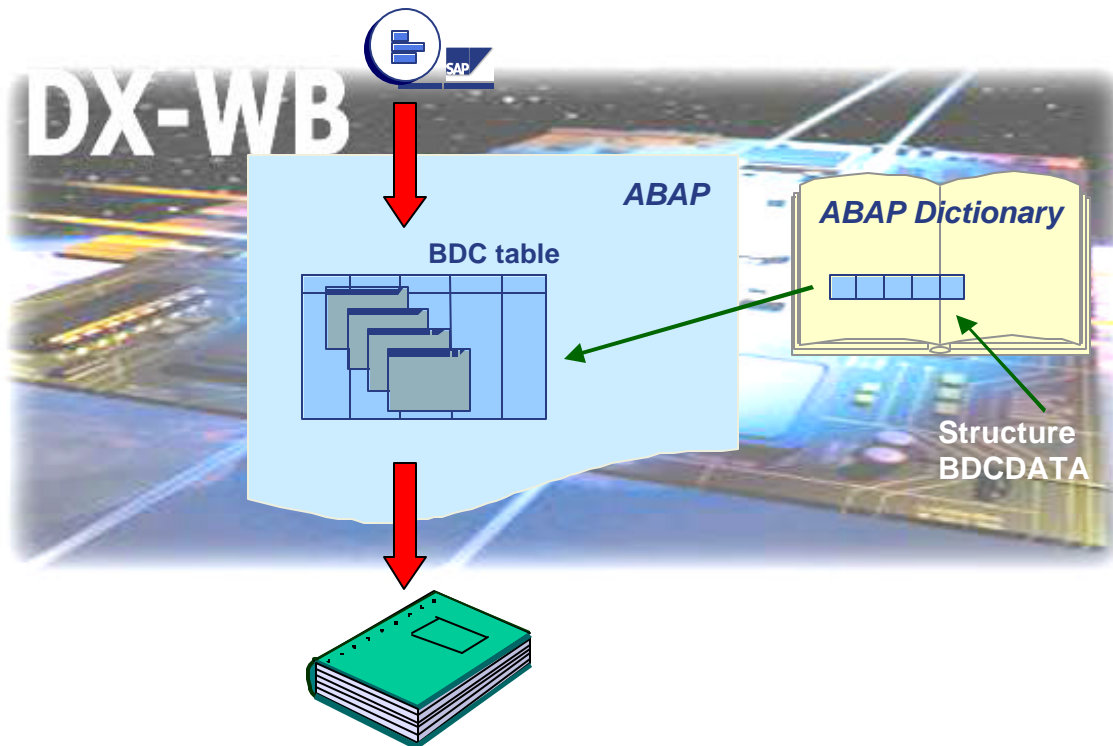


Batch input sessions

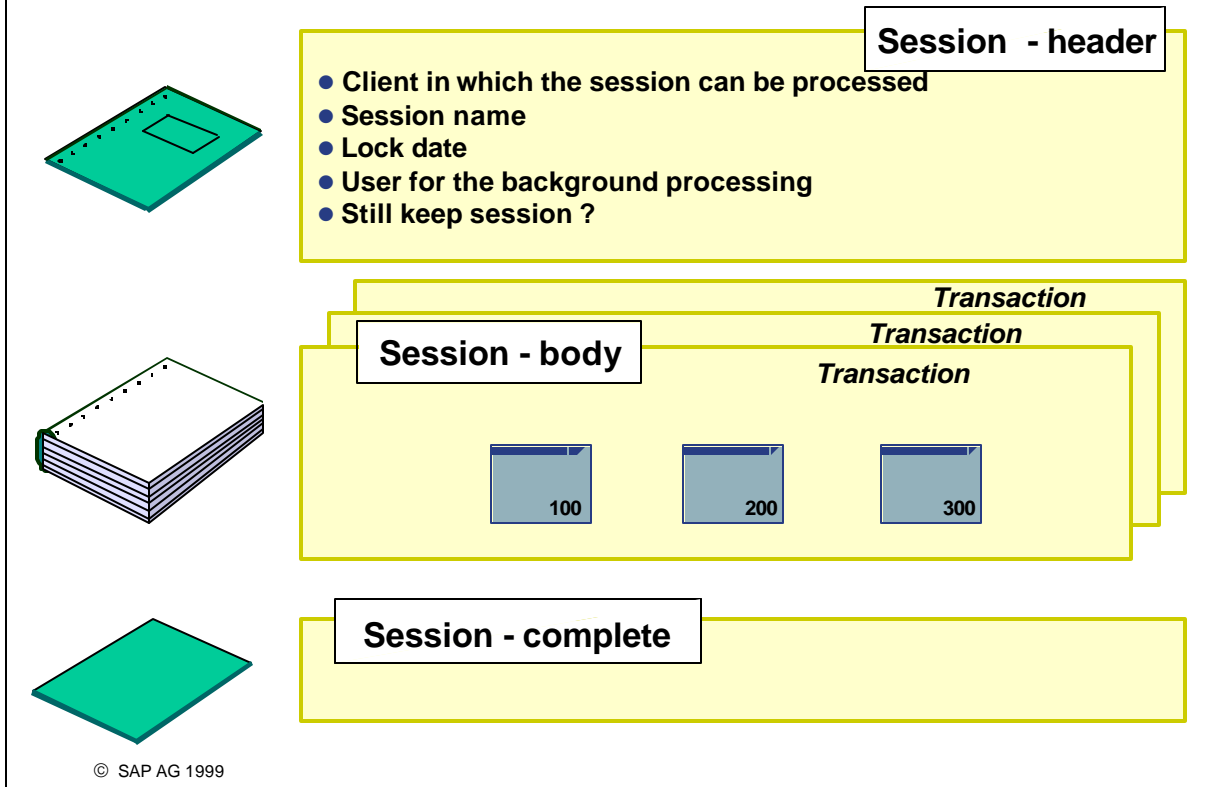
Batch input monitor



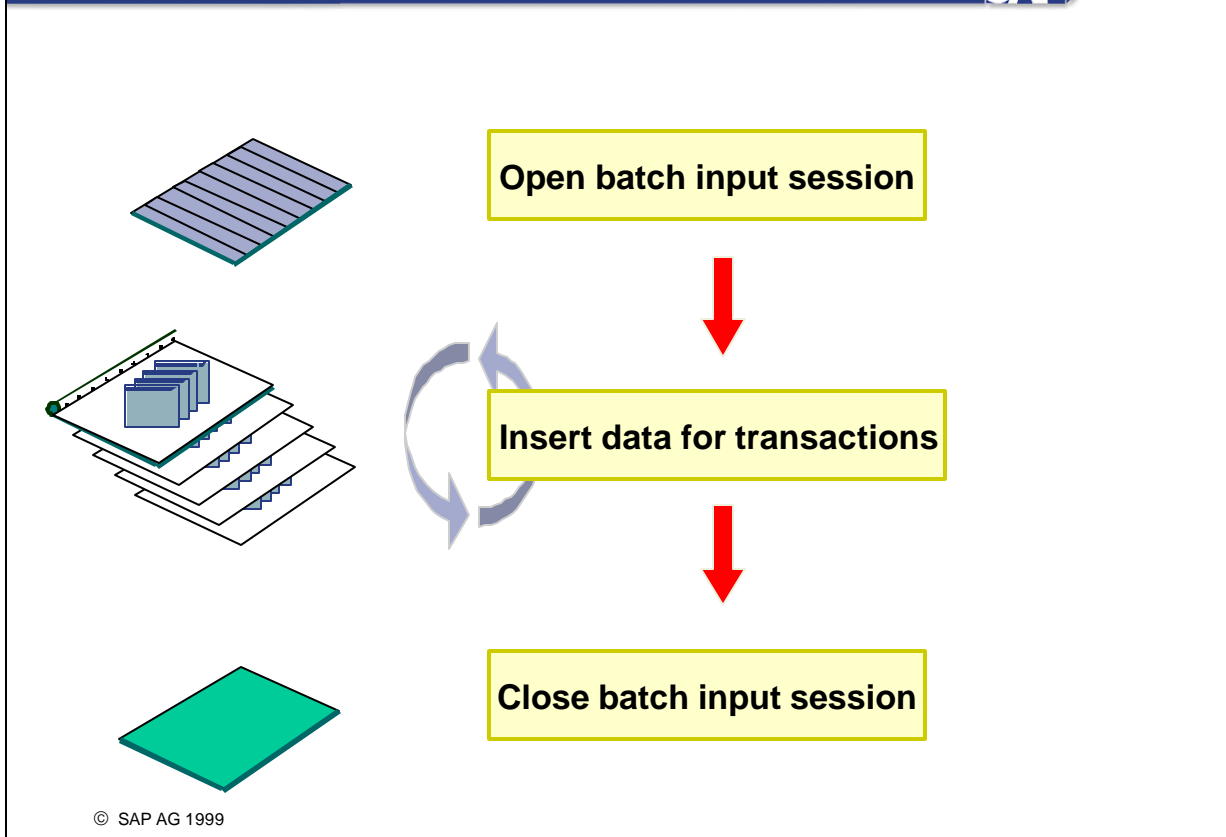
- Batch input, also known as Batch Data Communication (BDC) is an automated process used for dialog-free transfer of data to the SAP System.
- In order to ensure the same checks and updates are executed as in dialog processing, a **user dialog** is **simulated**.
- The central component of this method is the batch input session.
- These sessions are created using a batch input program and contain all data necessary to simulate the user dialog.
- After the batch input sessions are created, you use the batch input monitor to process the sessions. During processing, the online transactions are used to transfer the data to the R/3 database.



- The data is transferred into the session by batch input programs. For standard transfer these programs are provided in the DX-WB and for individual transfers they have to be created.
- Batch input programs have the following functions:
 - They provide structured work areas for the data to be transferred in the form of an internal table (BDC table).
 - They import the data.
 - They put the imported data into the BDC table.
 - They transfer the filled BDC table into the session.
 - The BDC table has always the same structure and contains the relevant transaction data for the data transfer.



- A batch input session consists of a header part (session header described by the ABAP Dictionary structure APQI) and a data part (transactions described by the ABAP Dictionary structure APQD).
- The session header contains general data about the whole session.
- The session body contains all the application-specific transaction data.
- The session complete closes the session; it can now be processed in the batch input monitor to transfer it into the R/3 database.



- Each batch input session must be opened and closed.
- The data for the application functions (transactions) is inserted into the batch input session.
- The batch input session contains screen data from individual transactions.
- Data is inserted into the batch input session for each transaction. One transaction is inserted at a time into the session header.
- A batch input program can create one or more batch input sessions in succession.
- Batch input sessions cannot be created in parallel

Structure of BDC Table

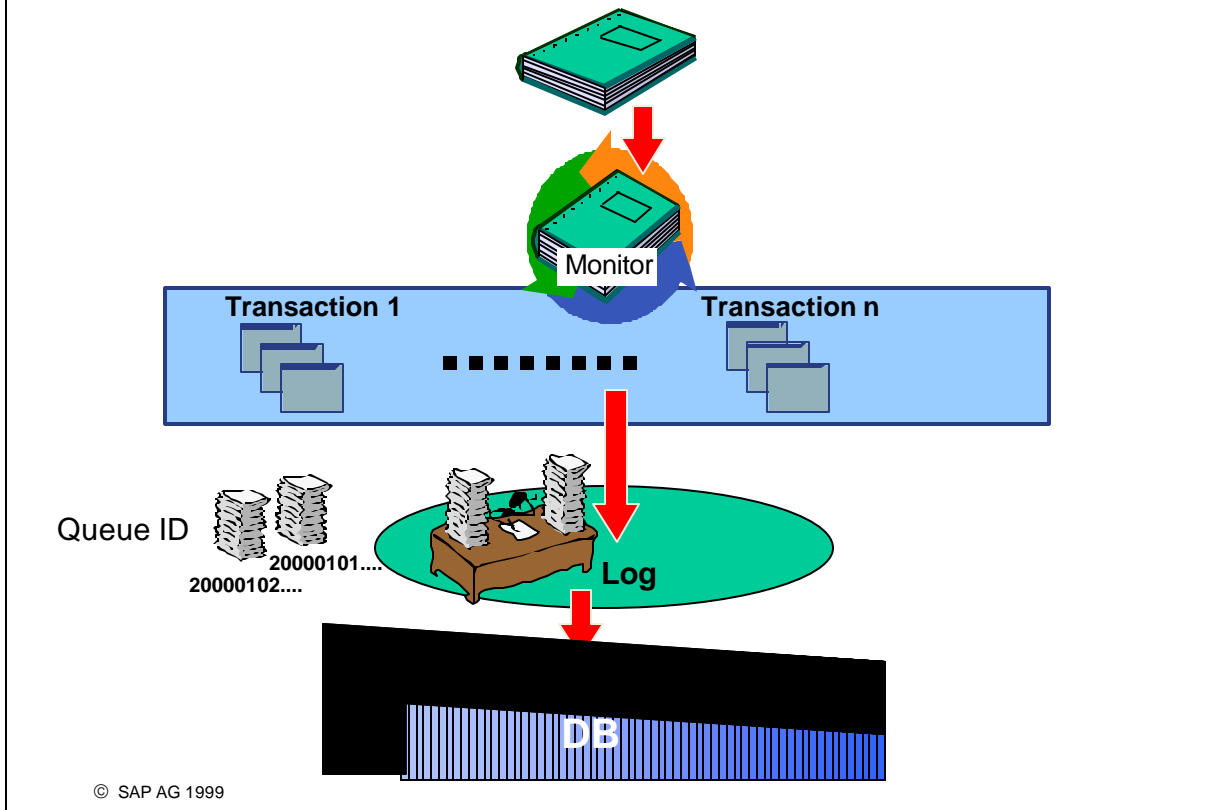


<i>Program Name</i>	<i>Screen No.</i>	<i>Start of Screen</i>	<i>Field Name</i>	<i>Field Value</i>
PROGRAM	DYNPRO	DYNBEGIN	FNAM	FVAL
SAPMF02D	0105	X		
			RF02D-KUNNR	Z-00-00001
			RF02D-BUKRS	0001
			RF02D-KTOKD	KUNA
			BDC_OKCODE	/00
SAPMF02D	0110	X		
			KNA1-ANRED	Mr
			KNA1-NAME1	Treusch
			KNA1-SORTL	Axel
			KNA1-NAME2	
			KNKA1-STRAS	Main St.
			KNKA1-PFACH	
			KNKA1-ORT01	New York
			KNKA1-PSTLZ	69190
			KNKA1-ORT02	
			...	

© SAP AG 1999

- The data is structured within a BDCDATA table according to the screens in the transaction. Each screen that is processed in the transaction must be identified in a BDCDATA record. This record uses the fields PROGRAM, DYNPRO and DYNBEGIN. A separate BDCDATA record for each value that can be entered in a field comes after this record. These records use the fields FNAM and FVAL.
- The field name for the OK code is always BDC_OKCODE for all screens. The field value of the OK code when a function key is pressed is made up of the backslash '/' and the number of the function button (e.g. '/11'). For other types of application actions (menu choices, pressing the SAVE icon etc.) the field value of the OK code is an '=' followed by the function code (e.g. '=save').
- The BDCDATA structure is defined in the ABAP Dictionary as follows:

Field Name	Data Element	Type	Length	Short Description
PROGRAM	BDC_PROG	CHAR	40	BDC Module pool
DYNPRO	BDC_DYNR	NUMC	4	BDC Screen number
DYNBEGIN	BDC_START	CHAR	1	BDC Start of screen
FNAM	FNAM_____4	CHAR	132	BDC Field name
FVAL	BDC_FVAL	CHAR	132	BDC Field value



- A batch input session gathers the individual screens of a transaction together.

You process these sessions using the batch input monitor. This process is the same as when you execute the corresponding application transaction on foreground; the data is passed through the screens and is then posted into the relevant SAP tables and the status records are written to the log file.

- The batch input function starts the application functions specified in that session (transactions identified by their transaction codes).
- The data from the session is then merged into the screens of the specified online transaction.
- With change transactions the data from the SAP database is first merged and then the data from the session is merged into the screens.
- With the batch input procedure data is always posted synchronously. With other procedures, which will be described later, data can also be posted asynchronously.

- The batch input monitor can be called from the system menu.
- The functions greyed out are only available when batch input sessions are processed in the foreground.

- The batch input monitor provides a range of functions for processing the sessions.
- Sessions can be processed further with the batch input monitor. You can get information about the status, screen sequence, screen contents, processors (users), processing mode, server and which user has been used for the user authorization.
- The system always flags transactions that cannot be processed because of errors. These transactions remain the session (error session). You can correct the errors and reprocess these transactions.
- The functions can be accessed from buttons/menus in the monitor.

- On the initial screen of the batch input monitor you can select and process single sessions or look at the logs of sessions already processed.

- The following processing modes can be selected:
- **Processing mode:**
 - Process in foreground: transactions run in a foreground dialog (interactive processing of the session).
 - Display errors only: transactions run in the background. If an error is encountered, the appropriate screen is displayed in the foreground for user correction.
 - Background: transactions run in a background process.
 - Target system: optionally allows you to specify the target computer for background processing of the session. If you do not specify this parameter, the system will choose an application server based on automatic load balancing (recommended!).
- **Additional functions:**
 - Extended log: W”, “I” and “S” messages can also be recorded in addition “E” messages. You can also use this option when processing files in the background.
 - Expert mode: the expert mode only works in interactive batch input. It switches the error message 344 off - that is, you can execute additional functions in batch input without getting the message each time - *“Batch input data is not available for this screen“*.
 - Screen standard size: automatic creation of screen standard size when processing in dialog (relevant for table controls or step loops).

- Additional session functions are available when processing batch input sessions in the foreground. These can also be called from the command line using special OK codes.

- If you are processing a session interactively, you can correct the transactions online or flag them as incorrect transactions. The corrections will be included in the session log.
- If you are processing a session interactively (process in foreground or display errors only), the ENTER button must be pressed to be able to switch between screens. On each screen you can correct data, or process the session using specific menu functions (see slide) or OK code entries.

- A session is assigned a status, displayed in the session overview, before, during, and after processing.
- The sessions are sorted by the date and time they were created and displayed according to status.
- In the overview, the following information is displayed for each session:
 - Session name:
Name of the session
 - Date and time:
The data and time the session was created.
 - Locked:
Session is locked until this date and cannot be processed until then.
 - Created by:
User who created the session.
 - Transaction and screen:
Number of transactions and screens.
 - Authorization:
The user whose authorization is used to process the session (only for background processing).

- Do not navigate to other transactions (using /nxxxx or /oxxxx while processing a batch input session.
- When processing in foreground, move from screen to screen by pressing *ENTER*.

- Statistics can be displayed for every session. This includes the total number of transactions in the session as well which transactions:
 - Have errors
 - Were processed
 - Were deleted
 - Still need to be processed

- A log is generated for every batch input session run.
- If there are errors during processing, the session is marked with status “Error”. If the session is processed again, an additional log is generated.
- All error messages from transactions in the session are logged in the session log. This log also contains batch input error messages generated due to problems during processing of transactions. The transaction error information includes the transaction code and the screen where the error occurred. The log also contains “Total” statistics.

- You can go to the session analysis even before you choose “Process / foreground”. This allows you to display the transactions and screens.
- After the session is run, you can display several different overviews of errors that occurred during the run as well as a list of the transactions not yet processed in the session.

- The session analysis is divided into three tabbed areas:
 - “Transactions”: Provides a list of all transactions in the session along with their status.
 - “Screen”: Displays an overview of the screens in each transaction. You can simulate (display) each screen including the data it contains.
 - “Log”: Provides a list of all session logs.

- The field list display under the “Screens” tab displays the screen data according to the BDCDATA structure.

- To choose “Screen simulation” (screen display) double-click a screen

- Each user can specify in the “default settings” his own basic settings for the batch input monitor. For example, only sessions with the prefix “BC420*” may be selected.

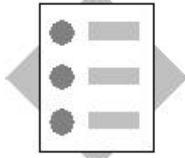
- Under *Utilities* -> *Export /import sessions* (in batch input monitor) session contents can be imported and exported at the operating system level.
- Further processing would be possible with an external software tool.

- Import and export are executed on the application server using physical or logical file names, which are set in R/3 Basis Customizing (transaction FILE).

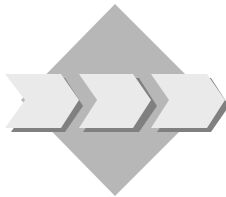


Unit: Batch Input Processing

Topic: Working with Batch Input Sessions



- Working with the Batch Input Monitor.
- Running of Batch Input Sessions.



The batch input session, which has been created out of the DX-WB, is run with the batch input monitor. This is how the legacy data are transferred to the R/3 system.

- 1 Run the BI session BC420SES-## (see BGR00 record layout).
 - 1-1 Import the BI session that you created before in the "Display all mode" in the R/3 system. (Name: BC420SES-## see BGR00 record layout)
While the system is running the session, change one address field of a debtor, e.g. change the place of residence.
Then switch the processing mode to "Display errors only".
 - 1-2 Analyze the log of the run and find out if errors occurred. Has the change of the debtor address been recorded during the run? _____
 - 1-3 Analyze the session statistics.
- 2 *Optional:* Repeat a BI session with errors.
 - 2-1 Start the run of the previous task again in the DX-WB:
A new file with debtor data must be read. On the selection screen, use the file BC420_DEBI_ERR.LEG as the. Use it to generate the debtors "Z-##-20001" bis "Z-##-20010", by changing the prefix on the selection screen. Create another BI session again.
 - 2-2 Switch to the BI monitor and run the session in the background. Wechseln Sie in den BI-Monitor und spielen Sie die.
 - 2-3 Analyze the log of the run. What kind of errors occurred?

Correct the errors by running the session again in the "Display all mode"!



Chapter: Batch Input Processing

- 1 Run the BI session BC420SES-## (see BGR00 record file).
 - 1-1 Change a customer location and switch to "Only display errors" during the BI session:
choose the path *"System->Services->Batch Input->Only display errors"* or enter the function code *"/BDE"* in the command field. All subsequent customers are processed in the background. You can see the result in the log.
 - 1.2 Field changes are logged. The session has the status "processed" after changes in foreground processing.

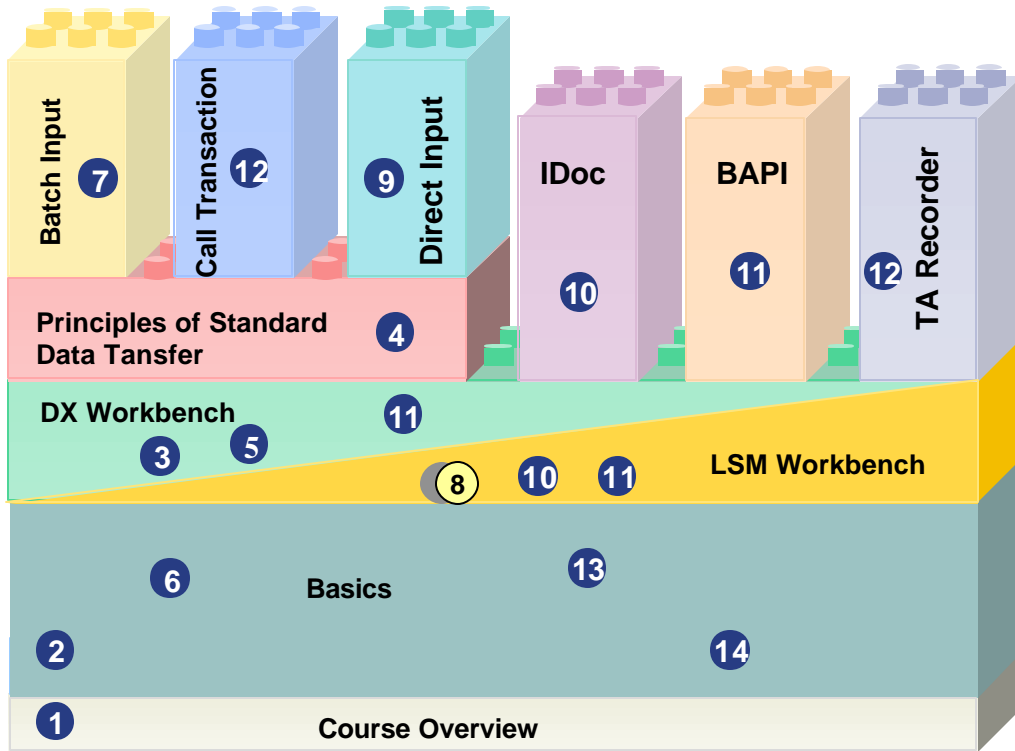
- 2 *Optional:* Repeat a BI session with errors.
 - 2-2 A BI session has errors.
 - a) The first customer in the old file had no customer number and was classified as with errors by the conversion program. See the error file (extension "ERR") This customer is not entered in the record file and the BI session.
 - b) The last (ninth) customer had no country in the address data. This record and its transaction are flagged as errors in the BI session.

 - 2-3 The session has the status "error".
The log contains: "Fill all required fields" "transaction error". You can display the filled fields of the screen in the screen view.

Contents:

- **LSMW Basics**
- **Defining Source and Target Structures**
- **Creating Field Mapping and Rules**
- **Reading Files**
- **Transferring Data to the R/3 System**

Course Overview Diagram





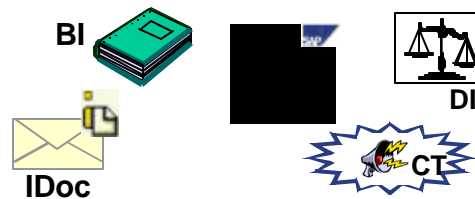
Basics

Structure Definitions

Field Mapping and Rules

Reading and Transferring Data to R/3

- Available as an add-on
- One-time and periodic data transfer from external system to R/3
- LSMW uses standard technologies

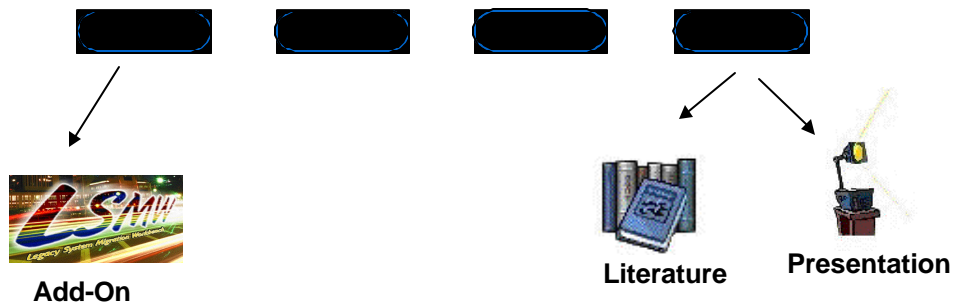


© SAP AG 1999

- The Legacy Systems Migration Workbench (LSMW) is an R/3-based tool that supports one-time and periodic transfer of data from non-SAP systems (legacy systems) to R/3.
- This easy-to-use tool supports the conversion of data from the non-SAP system that can then be imported into the R/3 System using batch input, call transaction, direct input, BAPIs, or IDocs.
- In addition, the LSMW provides a recording function, which you can use to generate a data migration object from a create or change transaction.



<http://sapnet.sap.com/lsmw>



© SAP AG 1999

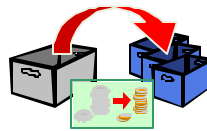
- The LSM Workbench is not a part of the standard R/3 System. It is available as an add-on for Release 3.0F and higher. If you are interested in the LSMW, contact SAP through one of the following channels:
 - SAPNet - R/3 Frontend: Component XX-LSM
 - E-mail: lsm@sap.com
 - Fax: +49-6227-742890
 - SAPNet: <http://sapnet.sap.com/lsmw>
- There you will find all available documents and information about LSMW as well as the LSMW software (transport file).
- The information available includes, among other documents:
 - A checklist for implementing the LSMW
 - A PowerPoint presentation describing the LSMW

- **Read data**

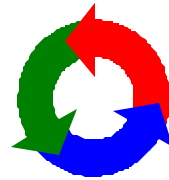
- Spreadsheet tables
- Sequential files



- **Convert data**



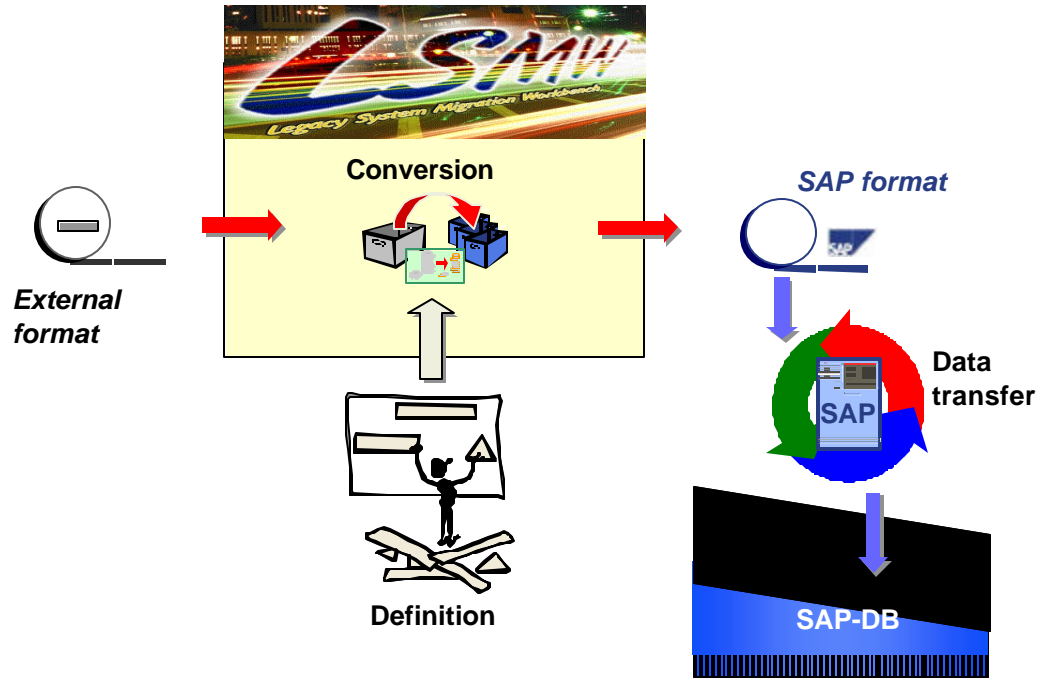
- **Import data**



Data transfer

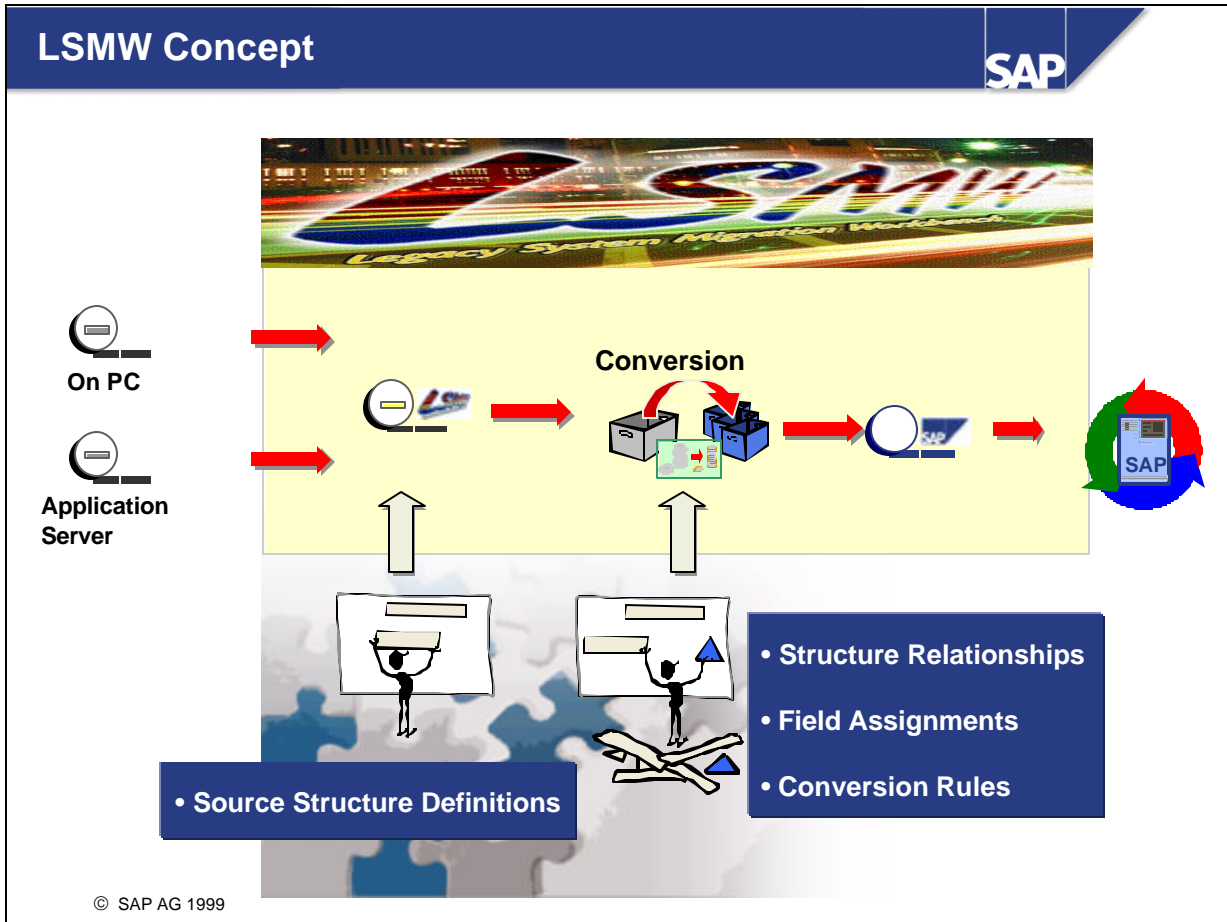
■ The core LSMW functions are:

- Reading data files from an external system into an internal LSMW format
- Converting the data, that is, formatting and assigning the data to the required structures
- Calling the standard transfer program or BAPIs or IDocs to perform the data transfer into the R/3 tables.

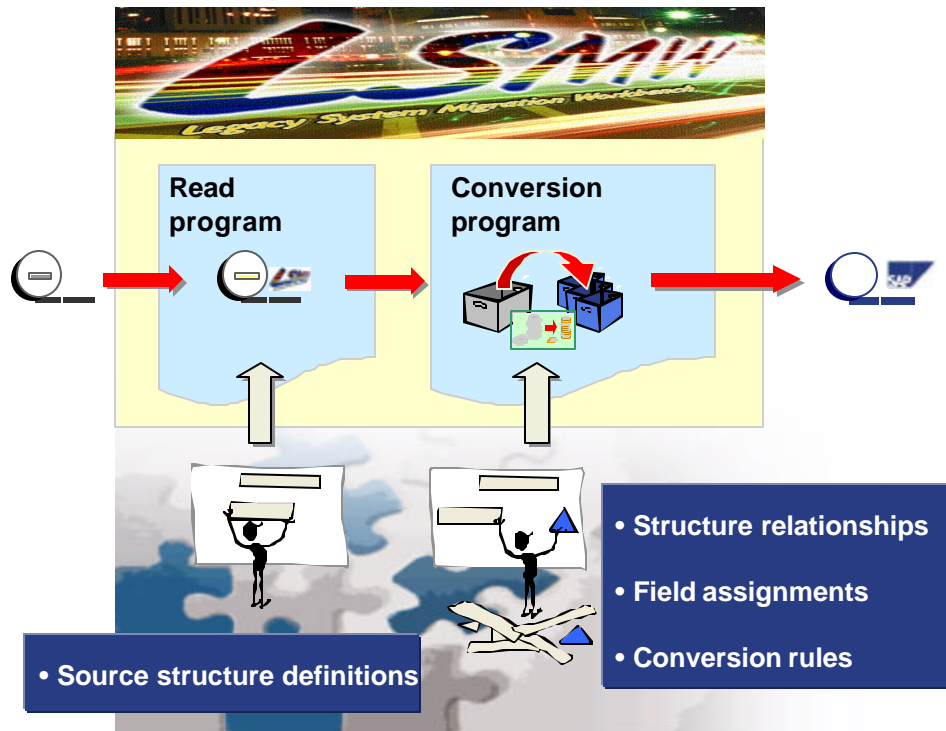


© SAP AG 1999

- The LSMW reads the external files and converts the contents into the corresponding target structures or fields, producing an output file in the required format. This output file can then be input to the appropriate data transfer program.



- Using the LSMW, you can read data from presentation servers (PC) or application servers. To read these files, you must define the structure of the file (source structure).
- The file contents are then read into an internal LSMW format and stored as an LSMW file.
- The next step is to define structure relationships and field assignments. These specify the mapping of the **source structure** fields to the **target structure** fields.
- In addition, you must define a conversion rule for every source field that maps to a field in the target structure. To do this, you specify how the value of the source structure field is to be transferred to the field in the target structure.
- Using these definitions, the contents of the LSMW file are converted to the target structure and can then be transferred using the specified transfer method.



© SAP AG 1999

- The LSMW generates two executable ABAP programs from the structure definitions and the conversion rules - a READ program and a CONVERSION program.

- The LSMW supports the following transfer methods:
 - Standard transfer programs used with batch input, call transaction, and direct input.
 - Data transfer using IDocs.
 - Data transfer using BAPIs.
 - Creation of a recording using the transaction recorder and generation of a batch input session.

- To start the LSMW, call transaction **LSMW**.
- On the initial screen, you can create related subprojects and objects by choosing *Create new projects*.
 - **Project:** Maximum of 10 characters to identify your data transfer project. If you want to transfer data from several external systems, you can create a project for each external system.
 - **Subproject:** Maximum of 10 characters; used as an additional subdivision.
 - **Object:** Maximum of 10 characters; used to identify a business data object.

- The following functions are available on the initial screen of the LSMW:
- *All objects:*
Creates an overview of all available projects.
- *My objects:*
Displays an overview of all objects you created.
- *All objects in a project:*
Displays a tree structure with all objects contained in the selected project.
- *Project documentation:*
If documentation was created, this displays all the documentation for the individual dialog boxes and steps. You can print and send the documentation, or also save it various file formats.

- A total of 26 different work steps are provided. The work steps displayed are **dependent on the selected object**.
- You can create a customized **personal menu** of the work steps. In the personal menu you can select a subset of the 26 work steps you want to display.

- First you must maintain the object attributes.
- You can specify an object ID. Name the object. To add the project to the list of all projects you have created, specify your name in the *Owner* field. This will then appear under *My objects* on the initial screen.
- Select whether the data transfer should be executed only once or periodically.
If you choose periodic data transfer:
 - PC files cannot be read
 - An additional step appears; the main program for periodic data transfer

- Choose the object type and import method. F4 *Possible entries* help is available for the input field.
 - For batch input, call transaction, and direct input, program documentation is available under the program name (Choose the *Glasses* icon).
 - If you have selected batch input recording, you can add additional recordings by clicking the arrow.
- Caution: If you choose the import method BAPI or IDoc, the system checks, when saving, whether a partner profile already exists for the preset partner (see *IDoc* unit) and the selected message type. If not, the system attempts to create a profile.

- To create a list of information about the points mentioned above, choose *Object overview*.

- Before you start using the LSM Workbench, create a “**mapping on paper**” for the object. For this, create and print an “object overview”.
- At this time, the overview only contains the design and description of the R/3 structures and the fields they contain. You can use this overview to help you map these target structures and fields to the corresponding source structures and fields.

- The file with the data from the external system must be imported by the LSMW. Therefore you have to define the structure of this data in the LSMW. It is called the **definition of the source structure**.
- The source structure may be structured in different ways.
- If the source structure is a flat structure, each line in the file has the same structure. For the definition the structure name and the field names as well as the technical characteristics of the fields must be defined.

- If the file consists of two or more different record layouts but in a fixed pattern (e.g. header then two details, header then two details, etc.), you must create a different source structure for each record type. The definition must include the structure name, the field names, and the technical field attributes.

- If the file consists of two or more different record layouts with a random pattern, then the first position in each row must contain information describing the record type. This is called the **ID field**.
- The definition must include the structure name, the field names, and the technical field attributes. In addition, when defining the fields, the ID field must be identified as such.

- In this step, you define the object structures, including name, description, and hierarchical relationships.
- In the dialog box, choose *Change*. You can now choose to create new structures, or change, rearrange, or remove these. Use the relevant pushbuttons to execute these functions.
- If you are creating more than one structure, a dialog box appears asking you to define the relationship between the structures; are they equal (flat structure) or unequal (hierarchical)?
- **Caution:** You can only create one structure per recording for migration objects created through a recording of a transaction as there is only one flat target structure available per recording.

- In this step, you create and maintain fields for the structures created in the previous step.

- When the data is imported you can specify how the data is converted:
 - Date values are converted into the internal date format (YYYYMMDD)
 - Amount fields are converted into the calculation format (1234.56, that is no grouping characters, only the decimal point). This format can only be used for amounts with two decimal places.
 - Packed fields with and without decimal places. The number for PAC"X" specifies the number of decimal places.
- Under *Selection parameters for importing/converting data* a flag can be set for fields of structures from the highest hierarchy level. If you set this flag, the field in question is provided as a selection parameter when you import and convert the data. This is usually used for tests.

- The structure relationships define the relationships between the source and target structures.
- The possible target structures are set when the object type and import method are selected.
- Usually there are target structures that must be selected (“required segments”). The system informs you of this with the message “This structure must be selected”.
- To set structure relationships, place the cursor on a field in the R/3 structure or target structure. Choose *Relationship*. A dialog box appears with a list of the source structures you have created. Select a structure from this list.
- If you want to change the relationship, you must first delete the existing relationship using the relevant pushbutton.
- A check function is available to check the structure relationships for errors. Messages for this check are displayed in the status bar (error message or “No structure relationship errors found”)
- **Note:** Many BI, CT, and DI programs use control record BGR00 or BI000. Always assign the header structure of the source structure to this record.

- In certain circumstances you may want to assign several source structures to one target structure. In this case proceed as follows: Create your target structures. Then assign the subordinate source structure to the target structure. Then the fields from both source structures are available for the fields of the target structure.

- In this step, you assign source fields to the target fields and specify how the field contents are to be converted.
- The system displays all fields for all target structures you selected in the previous step.
- The following information is displayed for each target field:
 - Field ID
 - Any source fields assigned
 - Rule type (default value, translation, and so on)
 - Code
- **Note:** Some fields are prefilled by the system; these are “technical fields” with the “default” rule type. Changes to the default may severely affect the data conversion process. You can reset a value to default that you have (accidentally) changed by choosing *Extras* → *Restore default*.

- The defined rules can be differentiated by their rule types. These rule types generally have the same characteristics.
- The different rule types are shown above.

- Reusable rules are those that are available to all parts of a project, and can be used in all objects in a project. Reusable rules are: *Default values*, *translations*, and *Own routines*.
- When you are assigning a reusable rule to the target field, the system suggests between one to three names.
- These will be names from one to three of the levels: domain, data element, or field level, which are distinguished as follows.

If the rule is set on the:

- Domain level, this rule is suggested when the target field is defined using this domain.
 - Data element level, this rule is suggested when the target field is defined using this data element.
 - Field level, this rule is suggested when the target field has the same field name.
- It usually makes sense to accept the system's proposal, unless the domain is very general, such as CHAR1.
 - This naming procedure keeps the number of conversion rules low and ensures consistency in the data conversion.

- **Assigning source fields:**
To assign a source field, place the cursor on a target field in the tree structure and choose *Assign source field*. A list of available source fields appears, from which you can select the desired source field by double-clicking it.
- **Deleting source field assignments:**
To unassign a source field, place the cursor on a target field in the tree structure and choose *Remove source field*. If only one source field is assigned, this assignment is removed. If more than one source field is assigned, a list of all assigned source fields appears; to select the desired field, double-click it.
- **After you have assigned the source fields, you define the conversion rules. The default is “MOVE”.**
Various standard techniques are available by choosing the relevant pushbutton.

- In this step, code assigned to the target field is deleted. In addition, source fields assigned to the target field are deleted. The target field then contains the following value:
 - For standard BI, CT, and DI: NODATA character (set in session headers such as BGR00 or BI000)
 - For batch input recording: NODATA character “/”
 - For BAPIs and IDocs: Character fields → Empty field ; Numeric fields → “00...0”)

- The target field is assigned a constant.

- Data is transferred using the ABAP command “MOVE”. For source fields not of type “C” or “N, this means:
- **Packed field:** Unpack in target field WRITE...TO...
- **Date field:** If target field is at least ten characters: Output format is set according to the settings in the user master. Otherwise, the data is left in the internal format; for example, 01.10.1998 (YYYYMMDD)
- **Amount field:**
 - For batch input and direct input: The amount value is formatted according to the settings in the user master.
 - For BAPIs and IDocs: The amount value is left in the internal calculation format.

- *Fixed value*: Suggested names for the fixed value are provided; these suggestions are domains, data elements, or field names.
- Choose a suggestion or enter a new name.
- You can specify the field description, length, type, value, and whether uppercase or lowercase must be used.

- Using translation, you can convert field values in the external system to valid values in the SAP System.
- Translation: The system displays *Possible entries* for the name of the translation; these are domains, data element names, or field names.
- Choose one of the *Possible entries* or specify a new name.

- To create a translation, you must define the source and target fields. Then, you must specify how the conversion is controlled.
- You can choose 1:1 translation or interval translation.

- You use this to set the type of translation. You can specify which translation table the system should search first for a value, and which alternative should be selected if no matching entry is found.

- Here you specify which value (old value) in the non-SAP system is to be replaced with the new value.
- **Note:** Only those values are converted for which the OK flag is set.

- Conversion value interval: You use this to set what the values in the non-SAP system (old values) are to be converted to in the SAP System (new values). Specify the value table with the value intervals to use for value conversion. You can upload the values from a PC file (text separated by tabs) to the table. F4 *Possible entries* help is available in the *New value* column.
- **Important to note:** During the conversion, the system only takes those values into account where the OK indicator is set.

- You can specify a prefix of your choice; this will be placed in front of the contents of the source field.

- You can specify any suffix you want to add to the field contents of the source field.

- Concatenation: You can combine two or more source fields.
- Note: All source fields involved in the concatenation must previously be assigned to the same R/3 target field before you can combine those source fields.

- Transfer left-justified: When this is set, the field contents are transferred left-justified.

- You can add your own ABAP statements for field conversion to the program. Your statements are then processed for every data record.

- You can include a subprogram and a subprogram call. The subprogram can include your ABAP statements for field conversion and will be executed for every data record.
- Note: You can reuse subprograms.

- Under the menu entry *Field mapping and conversion rules* you can get an overview of all the defined reusable rules.
- These are displayed in three categories:
 - Fixed values
 - Translations
 - User-defined routines

- In this step you specify all the files that are to be used in the following steps:
 - Your legacy data on the PC and/or R/3 server
 - The file for the imported data
 - The file for the converted data

- Enter the file path, file name and file description.
- Enter in "Code page ID" the ID of the character set in the external system.
- Determine the technical record description and the separator.
- Note: In the R/3 System the user ID <sid>adm appears and not in the operating system. You must ensure that the read or write authorization is in the selected directory.
- A file that contains data for several source structures can be assigned to several sources structures.
- file that contains data for only one source structure, can be assigned to only one sources structure.

- The system proposes a name for the file containing the imported data. You can change the name if you wish to.

- Depending on the object, you can define either physical or logical path/file names.
- The fields “Logical path” and the field “Logical file name” can only be filled, if the batch input or direct input programs called later support this. You can use F4 input help for both fields.
- Note 1:
You can freely assign path and file names within the framework of the operating system conventions.
- Note 2:
If your data is in several sets of files, you can use a wildcard (*) in the file name. You can specify the possible values for * under “Values for wildcard”.

- Example for use of wildcards in file names: Let us assume the legacy data is contained in the following four files:
 - File 1: D:\Mig\Purchase Orders\PO Header 1.txt
 - File 2: D:\Mig\Purchase Orders\PO Position 1.txt
 - File 3: D:\Mig\Purchase Orders\PO Header 2.txt
 - File 4: D:\Mig\Purchase Orders\PO Position 2.txt
- These are paired into two sets (*1.txt and *2.txt). File 2 contains the position data for the header records in file 1, and file 4 contains the position data for the header records in file 3.
- During data import, first import file 1 and file 2, and then file 3 and file 4.
- Use the wildcard "*" in the file names and define the values 1 and 2 for the wildcard.
- Note: You can also use a wildcard in the names of the files containing the imported and converted data.

- In this step you assign the defined data to the source structures:
- Note: If you later change the file name or the file characteristics, the file assignment remains the same.
- A file that contains data for several source structures can be assigned to several source structures.
- A file that contains data for only one source structure can be assigned to only one source structure.

- You have the following options with the function “Read data”:
 - If you want to process all the data belonging to an object, execute the function.
 - If you simply want to migrate a part of the dataset, you can limit the amount of data that you migrate in the field “Standard selection parameters”. Select your data in “Transaction number” “from ... to ...”.
- If you have selected one or more source fields as the selection parameters when defining the source fields, these fields are also available as selection parameters.
- Two checkboxes are provided for you:
 - Amount field: Amount fields are converted into the internal ABAP format (with a decimal point).
 - Date values: Date fields are converted into the internal ABAP format (YYYYMMDD).
- If you are using a wildcard in the file name of the input files and you have defined at least one value for the wildcard, you will also get a selection parameter for the wildcard. If you do not specify anything, all the defined wildcard values are processed.

- In this step you can display a part of or all of the imported data in a table. By clicking on a table row (or selecting *Field contents*) you can display all the information in this row.
- With the function *Change display* you can switch between one line and multiple line display.
- The colors of the individual hierarchy levels can be displayed in *Display color legend*.

- This work step is similar to the work step “Read data”.
- The imported data is converted into the target file the using the defined rules.
- If you do not select any data, confirm by executing the process. Otherwise select the data in “from....to...” in the “Transaction number”. You can also select several transaction numbers.
- If you have flagged one or more source fields as selection parameters when defining the source fields, these fields are also offered to you as selection parameters.
- If you use a wildcard in the file name of the input files and you have defined at least one value for the wildcard, you will also receive one selection parameter for the wildcard. If you do not enter anything, all defined wildcard values are processed.
- Note: The system first checks whether the data conversion program is still up-to-date. If it is not, it is automatically regenerated.

- In this step, you can display part or all of the converted data in a table. To display information about a line, double-click it, or choose *Field contents*.
- To choose between a one-line or multiple-line view, choose *Change display*.
- To display information about the colors of the hierarchy levels, choose *Display color legend*.

- The steps depend on the selected object type:
 - Standard batch input or recording:
 - Create batch input session
 - Process batch input session
- Standard direct input:
 - Start direct input session
- BAPI or IDoc:
 - Start IDoc generation
 - Start IDoc processing
 - Generate IDoc overview
 - Start IDoc postprocessing

- If you have selected a standard SAP transfer program, it may support batch input, call transaction, or direct input or some combination of them.
- If you use batch input:
You must create the session and then process the session using the batch input monitor.
You can go to the batch input monitor; only the batch input sessions for the selected object will be displayed.
- If you use call transaction:
You can start the program directly and import the data immediately.
- If you use direct input:
You can start the direct input program directly (should only be used for testing), or start the data transfer using the direct input monitor.

- From the initial screen select *Goto ® Administration*. Here you will find an overview of all the existing projects.
- You can create, edit, display, delete, copy or rename projects, subprojects, objects and reusable rules.
- By double clicking on an entry you can go to the entry display.
- If you position the cursor on an entry, you can create a personal remark in the documentation. With each processing the name and date of the last change is saved.

- The following authorization levels can be assigned for the LSMW application:
 - Display
The user can display all projects with their work steps. User cannot change data.
 - Execute
The user can display, import and convert data.
 - Change
The user can “execute” and can also change and copy objects.
 - Administrate
The user can use all the functions offered by the tool.



Unit: LSMW

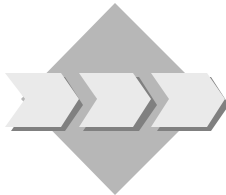
Topic: Conversion and Transfer of Documents Using the Legacy System Migration Workbench



- Creating a mapping plan for FI documents
- Defining translation rules in LSMW for the documents.
- Mapping of the document data using the LSMW.

Conversion of documents using the Call Transaction

Legacy FI documents should be imported into R/3 via the LSMW using the Call Transaction.



LSMW project: BC420-##

Subproject: DOCU-##

Object: CT-##

**File with documents in legacy format:
BC420_DOC_1_HEAD_POS.LEG**

The FI documents are available in the file BC420_DOC_1_HEAD_POS.LEG on the application server. This file contains five documents with three items each. This data should be transported into the corresponding SAP record layout using the LSMW. Transfer this file with program RFBIDE00 using the call transaction.

Part 1 of the Task

- 1 Perform the following test: Create an FI document online using transaction FB01.

Document header

Document date:	today
Document type:	SA
Company code:	0001
Currency:	UNI

First Document Item

Posting key:	50
Account:	100000
Amount:	200

Next Document Item

Posting key: 40
 Account: 113100
 Amount: 200
 → **Save document!**

LSMW-Activities:

- 2 Start the LSMW.
 - 2-1 Create the project BC420-## in the LSMW with the description "Financial Data".
 Create the subproject DOCU-## with the description "Documents".
 Create the object CT-## with the description "Documents with CT".
 - 2-2 Maintain the object attributes for batch/direct input:
 Object: 100 financial documents
- 3 Go to the object overview where you can display the objects in a table. Save the list as a local file in spreadsheet format. Assign any name to it and display this table in Excel.
- 4 Create a **Mapping Plan** for the transport of the legacy documents.
 With the course material you will find a printout of an extract of the Excel sheet.

4-1 The data from the legacy system is displayed in the following table:

Field name	Field length	Field value type	Description
HEAD-DATE	10	20.01.1999	Document date
HEAD-TYPE	4	07	Document type
HEAD-REF	9	123456789	Old document no.
HEAD-CU	2	\$ or DM	Document currency
POSI-KEY	4	0004 or 0005	Posting key
POSI-AMOUNT	15	1000,10	Amount in document currency
POSI-ACCOUNT	10	100000	Document account
POSI-TEXT	6	Info	Document item text

- 4-2 Which field, required for creating a document online, does not exist in the table? _____
- 4-3 To continue, use the value of the field in step 1.
 Value: _____

4-4 Create a mapping plan using the attached table. Complete the source fields: Field name, length, conversion method and coding/comment.

Tip: The document category (07) corresponds to the document type G/L account. The posting key (0004) corresponds to the debit posting and (0005) to the credit posting.

4-5 Check the mapping plan, comparing it to the solution (Excel sheet).

Part 2 of the Task

5 The source file consists of 2 structures:
The document header structure and the document item structure. Create the structure HEAD and the structure POSI one level below.

6 Create the source fields for the corresponding structures:
(the number in the brackets indicates the field length)

6-1 The source fields for the **document header** are:

- SET1(1) Set indicator ID for header : H (H for header),
- REF(9) Reference number in the legacy system,
- DATE(10) Document date **note: field type DDMY**,
- TYPE(4) Document type,
- CU(2) Document currency.

6-2 The source fields for the document item are:

- SET2(1) Set indicator for items : P (P for item -> means position)
- KEY(004) Posting key,
- ACCOUNT(10) Account number,
- AMOUNT(15) Amount in document currency,
- TEXT(6) Description of the document item.

7 Structure relationships:
BGR00 and BBKPF must be assigned to the document header. And BBSEG must be assigned to the document item.

8 Field Mapping:

8-1 Perform the mapping according to your mapping plan.

The following fields need to be assigned:

- Document date: Move
Note: The date is converted by the LSMW into the internal ABAP format. The user-defined date format must be set to the internal format YYYYMMDD or the following ABAP command can execute the formatting:
- BBKPF-BLDAT+0(2) = HEAD-DATE+6.
- BBKPF-BLDAT+2(2) = HEAD-DATE+4.
- BBKPF-BLDAT+4(4) = HEAD-DATE.
- Document type: Conversion
- Company code: Constant (0001)
- Currency key: Conversion (e.g. \$ -> USD)
- Reference document number: Move
- Posting key: Conversion (e.g. 0004 -> 40)
- Amount in document currency: Move
- Item text: Move
- Account: Move

- 9 Specify the following path under "Specify files":
The source file is BC420_DOC_1_HEAD_POS.LEG on the application server.
Select codepage 1100.
Also maintain the file name under *Legacy data* [on the R/3 server (application server)]
- In *File contents* choose *Data for several source structures (seg. File)*.
 - Select code page 1100.
- 10 Assign the files:
Document - file: BC420_DOC_1_HEAD_POS.LEG
Items - file: BC420_DOC_1_HEAD_POS.LEG
- 11 Import the legacy data now.
Display the converted data as structure and as field display. Check the data!
- 12 Convert the data.
Display the read data as structure and as field display. Check the data!
- 13 Perform the step *Start Direct Input Session*. Select the program RFBIBL00.
Use the Call Transaction technology.
- 13-1 Check the file first! If no cancellation situation is determined, the actual transfer can be executed.

13-2 Switch off the function *Check file only!* Start the data transfer. The data is transferred with the call transaction.

Write down the document numbers of the first five documents.

14 You can look at the documents using transaction FB03.

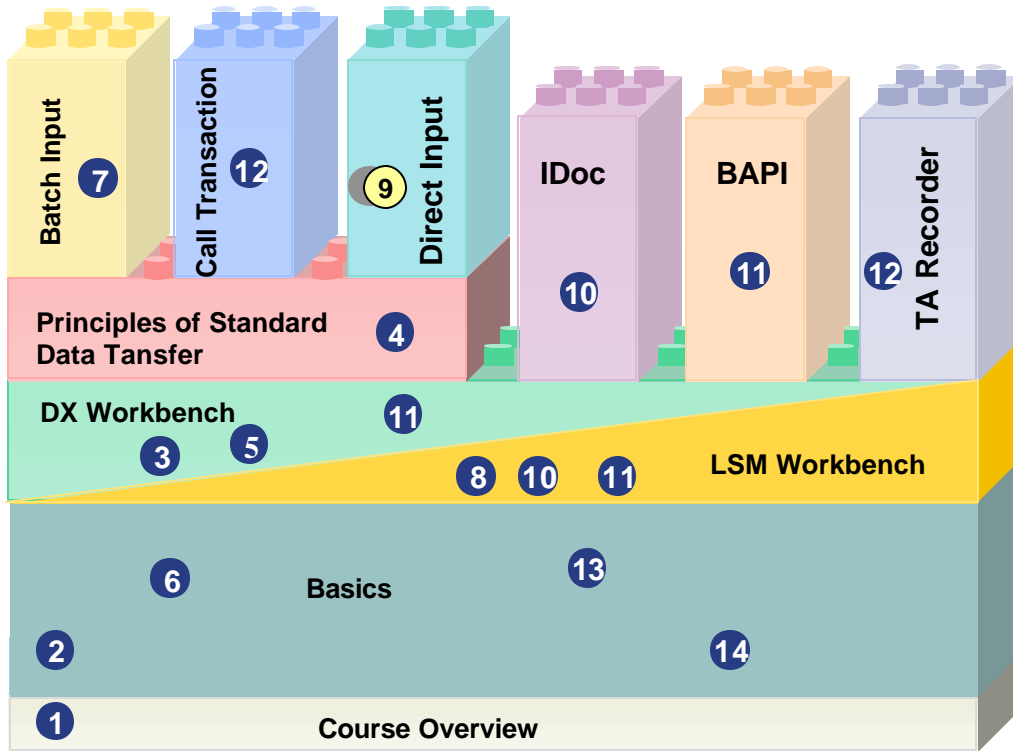
15 *Optional exercise:* The new file **BC420_DOC_1A_HEAD_POS.LEG** with documents in legacy format has been created. But in this file there is an error in the data record.

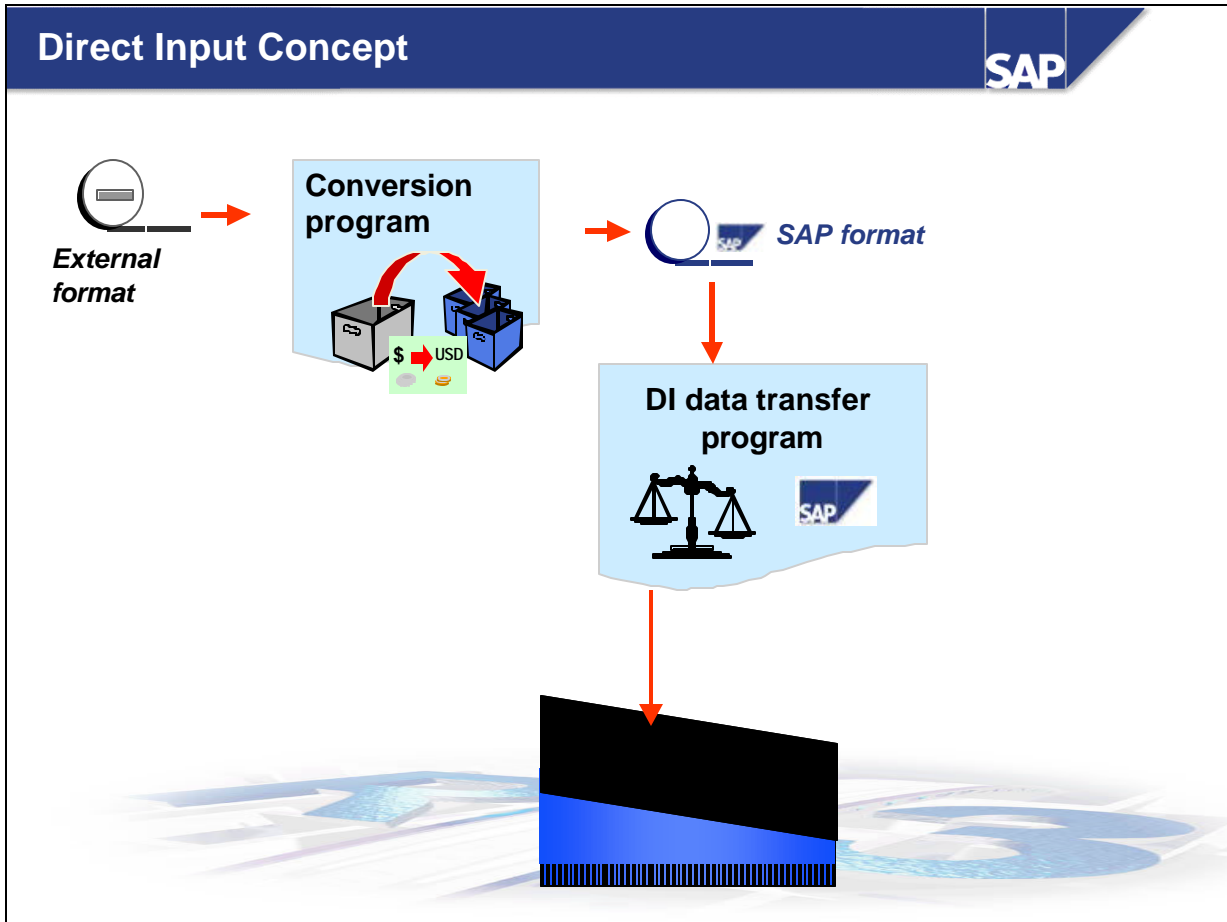
Perform steps 9 to 13 again with this file. When you transfer the data, you will receive an error message. This faulty data record is transferred into a batch input session. Process this session in the display all mode and correct the errors.

Contents:

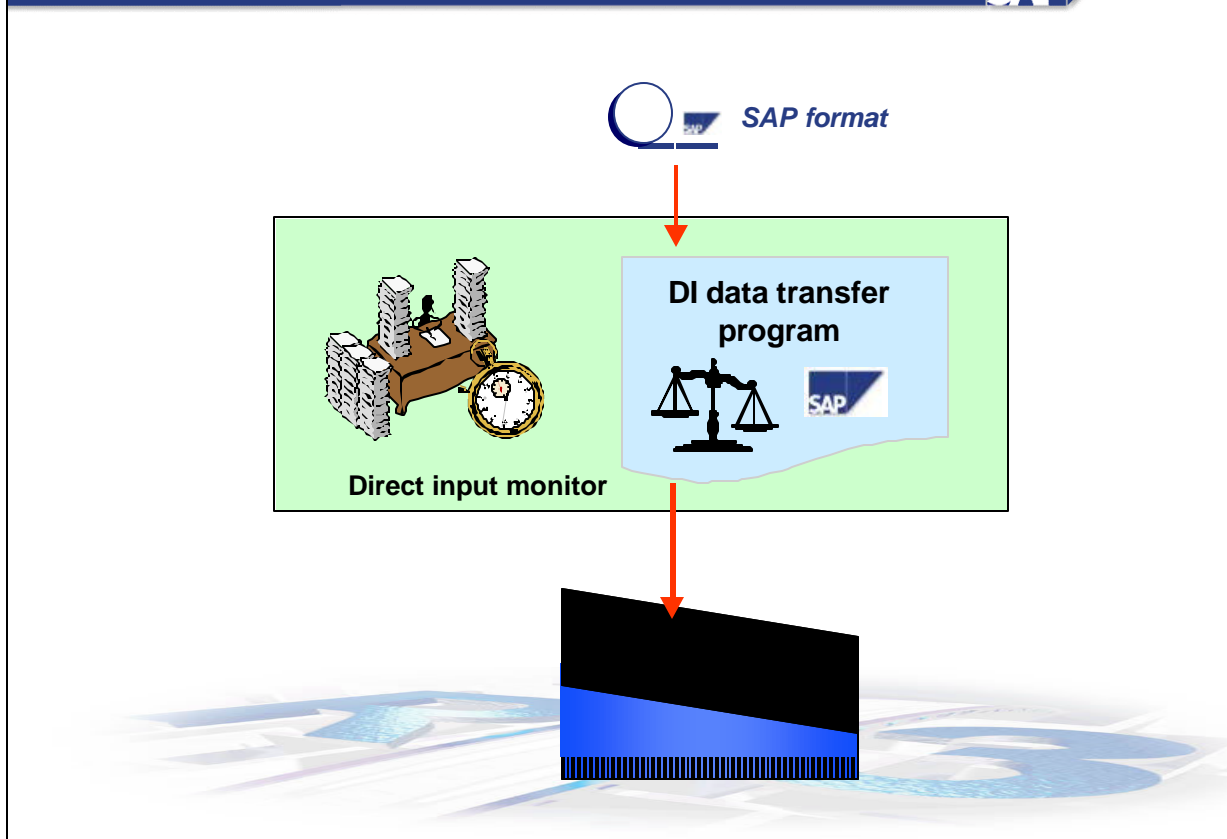
- Direct input concept
- Direct input monitor

Course Overview Diagram





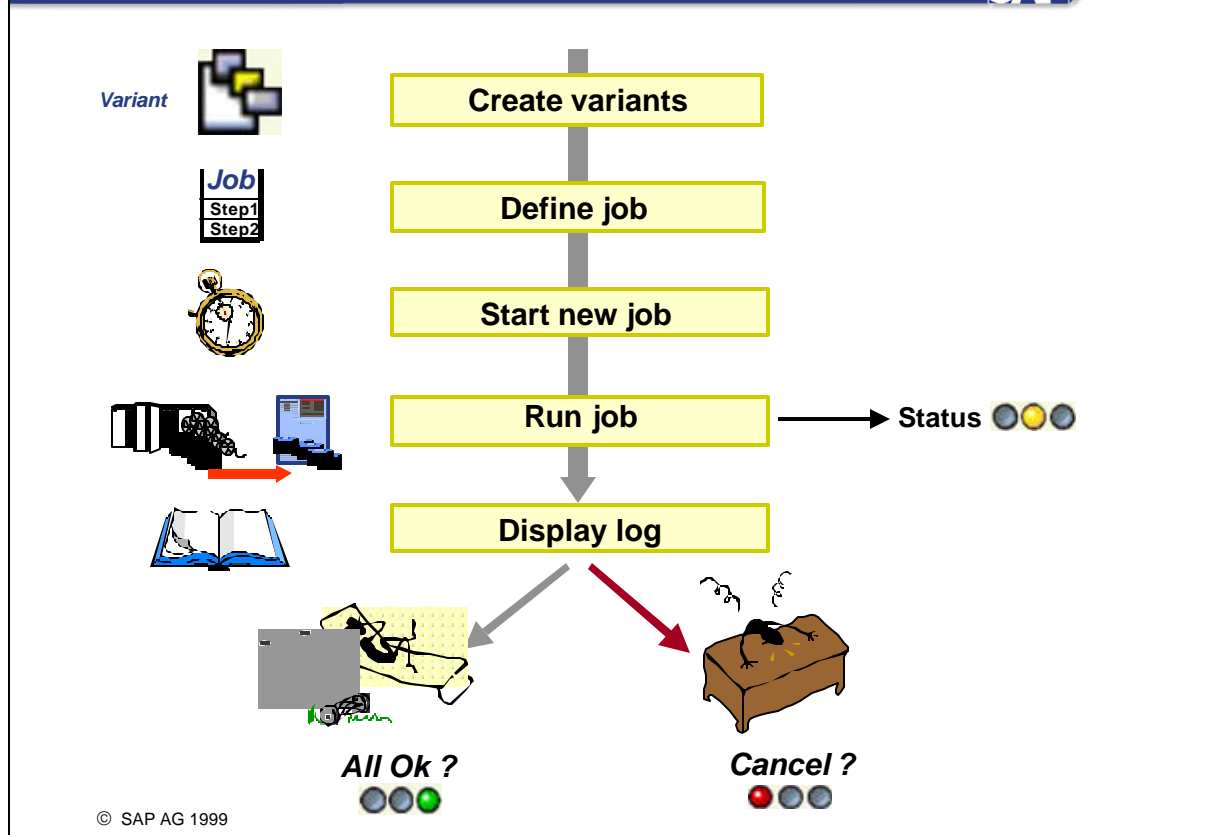
- An alternative to the batch input method is the direct input (DI) method. DI is more efficient than the other methods making it especially useful for transferring large datasets. Unlike batch input, no sessions are created - the data is updated directly. No screens are involved in this procedure. The data is entered directly into database tables by calling function modules that carry out the necessary checks.
- If case errors occur during the data transfer, direct input has a **restart mechanism**. To enable restart, direct input programs must be processed as background jobs run under the direct input monitor (program RBMVSHOW or transaction BMV0).
- If you are working with test data, you can start direct input in foreground. Make sure that neither error logs are created nor that the possibility of a restart exists in error situations.
- SAP strongly recommends that you use transaction BMV0 for the actual data transfer.



- Although you can test direct input in the foreground, it is strongly recommended that you schedule production data transfers using direct input in the background using the direct input monitor:

This ensures that in case of errors you can restart the data transfer program. This restart capability guarantees that no duplicate entries will be added to the database if the program is restarted at the point it terminated. A processing log is available to help you identify, correct and repost any errors.

- Using direct input in the background also has the advantage of better performance. .
- The following direct input programs are included in the direct input monitor:
 - FI documents (RFBIBL00)
 - Material master (RMDATIND)
 - Environment data (SAPLC131..132...133)
 - Sales documents (RVINVB10)



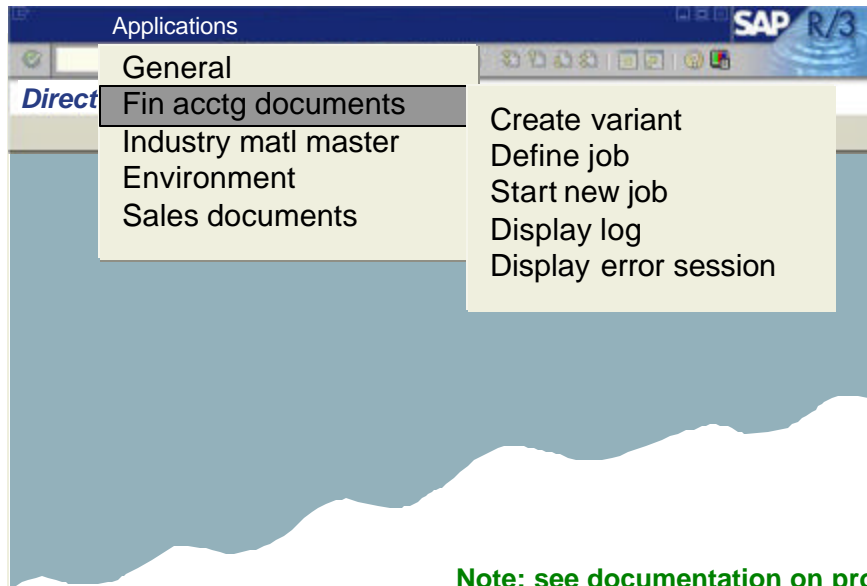
■ Direct input run:

- First create a variant for the transfer program.
- Define a job with this variant.
- Schedule the time of the job (start: immediately, date, time, by job, by event, etc).
- Analyze the job log after the job run.
- If the job is terminated, it can be reset.

☒Note: The restart mechanism will not work if the file contains formal errors (i.e. if the records are not in the SAP Record Layout). Use the “check file” function to ensure that no formal errors exist before processing the file using direct input..

Example: FI Documents

SAP



**Note: see documentation on program
BMVSHOW (BMV0)**

© SAP AG 1999

- You have to create a variant for each direct input program. The standard rules for creating variants apply.
- In the menu path *Job administration -> Define new job* you define control data for a job. A job is uniquely identified for each client by its name. You have to specify the report name (e.g. RMDATIND) and a variant name. If you want, you can also specify a server for the background job. This can be overwritten later. You can also specify a user name under which the direct job runs. This user requires the authorizations for all the applications.
- With the function *Start new job* start the job you defined earlier (F4 on first input field). You can change the default background server or user name. On the following screens enter the print parameters and start time.

Job Analysis **SAP**

The screenshot shows the SAP Job Administration interface. The title bar reads 'Job administration Applications' and 'SAP R/3'. Below the title bar, the text 'Direct input :administration' is displayed. Two buttons are visible: 'Job details' and 'Job log'. Two callout boxes provide details about these buttons:

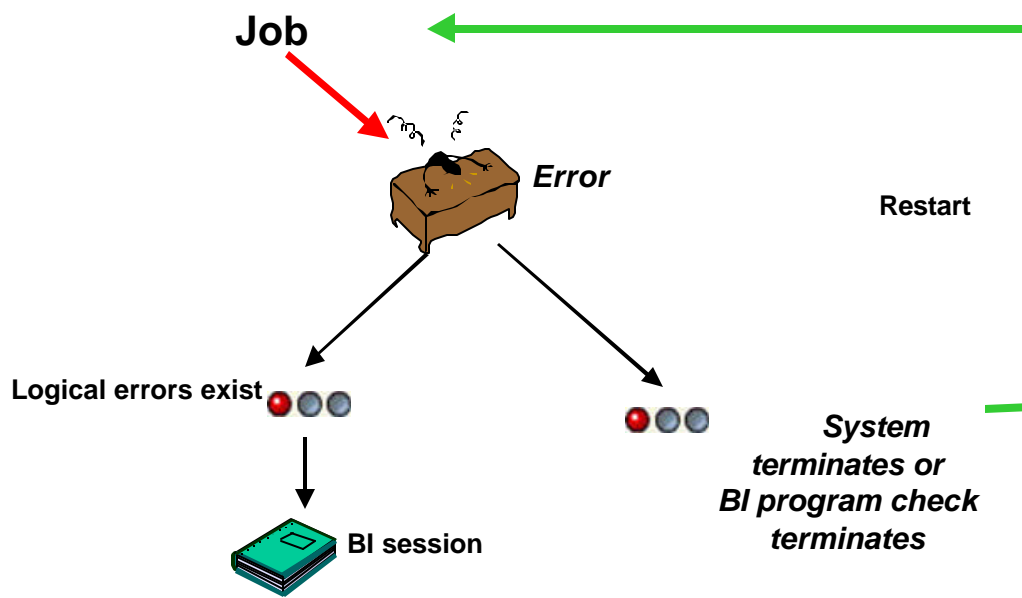
- **Technical information**
 - Program name
 - Variant name
 - Server
 -
- **Administration information**
 - Number of errors
 - Number of restarts
 -

● **System messages are listed chronologically**

● **Long text can be displayed**

© SAP AG 1999

- You can display details of the selected job in the detail display.
- By “updating” and selecting the job, the current status of the job is always displayed. If the job is terminated, say, due to a database error, the job is assigned status *Batch: job terminated*. In this case you can analyze the error (function *Display log*) and remove the error. Then choose *Job administration -> Reset job*. The job is restarted and the data transfer is continued from the point at which it had terminated.
- To assure data integrity, each program has a restart mechanism. Using some function modules the program writes synchronization information to table TBIST. If there is a termination, then it is guaranteed that the data transfer can be **reset to the correct place**. This means, for example, that no document is posted twice or not posted at all.
- System table TBIST contains entries:
 - Number of data records with errors
 - Last number processed when program terminated

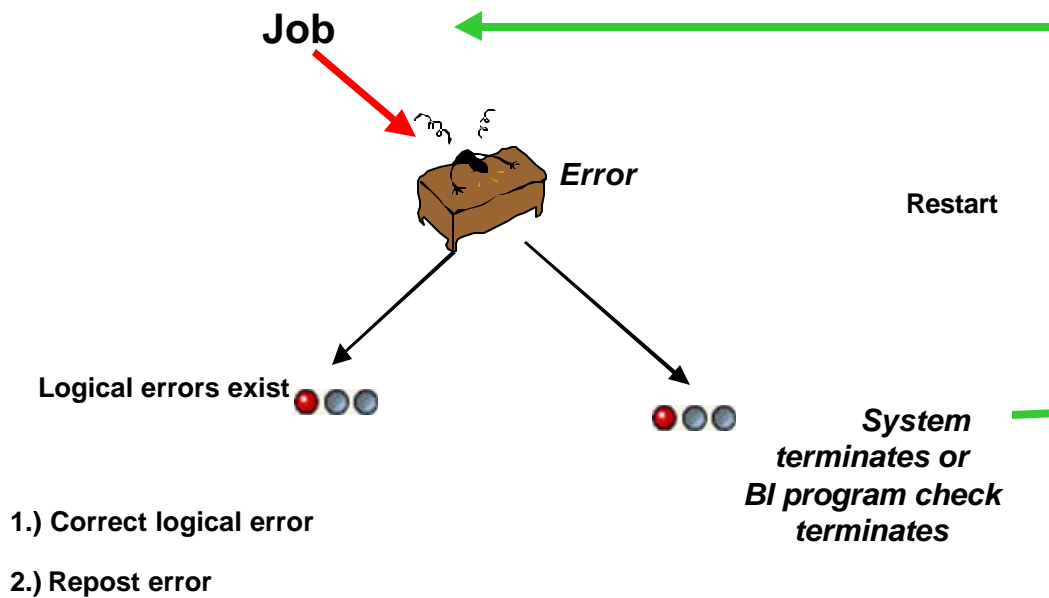


1.) Process BI session

2.) Change job status

© SAP AG 1999

- Each direct input program carries out checks before the data is transferred. If an error occurs (e.g. in RFBIBL00: data record has no posting key) this terminates the job without posting a single data record. The job is assigned the status *Background job terminated*.
- If the file only has logical errors - errors occur when the data is checked in the system (e.g in RFBIBL00: the currency amount is missing from the data record), this does not result in termination of the job.
The job is completed and is assigned status *Completed: job executed*; but it does contain the logical error. Records containing errors are placed in a batch input session.
- The session name of the first BGR00 record is used. Once the error has been corrected, you can process the batch input session and complete the data transfer. The status of the job with the function *error processing* should be set to: *Set to done*.
- For more information see the application help documentation for the FI documents under the DX-WB documentation.
- Note: The program RFBIBL00 can process only 20 data records without the BMV0 for the direct input/ call transaction technique. It should be executed within transaction BMV0 because of the restart capability.



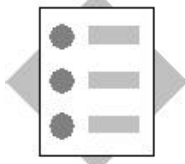
- An error file has been created for material master records which you can display and process in RMDATIND (or by choosing *Applications -> Material master industry -> Display logical errors*). After you have processed this file (or, for example, you have corrected a Customizing setting), you can start the error handling by choosing *Error processing -> Repost errors*. The data is not read from the original file, it is read from the processed file and the system attempts to repost the data after it has been checked.
- For sales documents: If errors occur when processing your job, you can find these errors in the SAP System using a detailed job log, correct them and repost the documents.

- If you want to use the direct input method for a periodic job, you must adhere to the following:
 - In the job definition, select “Periodic allowed” for the job.
 - You must schedule the job using the direct input monitor.
- If you want to add additional steps to the background job, save the start data for the job by choosing “*Start data transfer using direct input method*”. Choose a start time in the future. After saving, start transaction SM37 and edit the job step list. The job name here is the job name from the job definition.
- Make sure you check that the job has completed each time it runs. If the job terminates, the successor job will not start. Use the *History* function to check which jobs in a job definition are periodic jobs. If a job does not start because the predecessor job terminated, this is marked as “Invalid” in the history list.
- If you want to restart a job that terminated, first check to make sure the specified input file is still correct. The input file is usually switched each time the periodic job runs. If you restart a job that terminated, and the input file has been switched, data integrity will probably be affected.

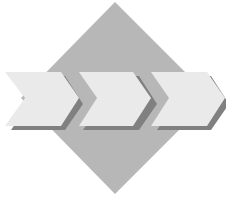


Unit: Direct Input

Topic: Transfer of FI Documents using Direct Input



- Use the direct input monitor for the data transfer.



Transfer financial accounting documents in the R/3 system using the Direct Input Technique.



Job: **Job-##**

Optional: As only one user can use the direct input monitor at a time.

To transfer the documents, do not use the Call Transaction – use the Direct Input method instead. You can use the same record layout structure. Hence you can use the project BC420-## with the subproject BELE-## and the object CT-## again.

- 1 Start the *Direct Input program* in the LSMW. Make sure to select the *transaction BMV0* for the transfer using Direct Input.
 - 1-1 Start the *Direct Input Management*.
 - 1-2 You will find the path for the Direct Input steps under: *Applications -> Financial Accounting Documents*.
 - 1-2-1 Create the variant *VAR##* for the program RFBIBL00. Specify the file name of the LSMW output file (converted data). Select the direct input technique for the transfer.
 - 1-2-2 Create the job *Job-##* for the program RFBIBL00 and this variant *VAR##*.
 - 1-2-3 Start the new job immediately and follow its progress. (Update the overview!)

Contents:

- **Principles of the TA Recorder**
- **Recording function in the LSMW**
- **Using the TA Recorder**
- **Principles of batch input and call transaction**
- **Integrating TA Recorder programs into the DX-WB**

- If the SAP standard program does not provide the functions you need for data transfer, you can use the transaction record.
- Note: For data transfer, the transaction (TA) recorder requires a “flat file”, that is, a file without any hierarchy or substructures.

- The TA Recorder enables you to transfer data into R/3 using batch input or call transaction.
- The TA Recorder can only work with a flat structure. Files with different record structures (for example, header structure) cannot be used.
- The TA Recorder is used in the LSMW to create recordings of transactions. Batch input sessions can be created from the recordings.
- The TA Recorder can itself execute a data transfer from the recording using batch input or call transaction.

- The TA Recorder can record transactions. The recorder must be activated before the transaction is processed. Then you can run the transaction as though you were processing it online.
- After the transaction has finished, the recording will contain all the screen numbers, field names, field values and the associated module pool and transaction code.

- In the above example the transaction *Change customer* FD02 is recorded.
- In addition to the screen fields, the fields BDC_CURSOR and BDC_OKCODE are also automatically recorded.
- **Note:**
 - **You must change the field values for all the fields required later for the transfer. For the recording a change can simply mean overwriting one single character in a field with the same character.**
 - **If fields are to be filled using SET/GET parameters, the value must also be changed.**
 - **With online transactions screen sequences may appear that are different than in the recording.**

- If there is a subscreen on the screens, this is automatically recorded. In the recording the field BDC_SUBSCREEN containing the name of the subscreen appears.
- With customer master data the address is a subscreen area. The central address management is integrated here. All the address input fields are displayed on one subscreen.
- Central address management is only used, if you specifically select this option on the initial screen.

- The result of the recording is a list of all the fields and field values that were changed in the recording.

- During a recording, the following additional fields are automatically recorded.
 - The *BDC_OKCODE* field contains the function code triggered on the screen.
 - The *BDC_CURSOR* field contains the field name of the last field the cursor was positioned in before exiting the screen.

- The information recorded in the TA recorder can also be displayed online for each transaction.
- To find out the current module pool name and the screen number choose *System® Status* from the menu.
- To determine the field name place the cursor on a field and select F1 help. The name of the screen field can be found under *technical information*.

- Some screens contain subscreen areas.
- To determine the field name, place the cursor on the field and choose *FI*. The name of the screen field is specified under *Technical info*.

- The tabstrip element shown is used on the *Maintain User Profile (Own data)* screen.
- Under the tabstrip title *Defaults*, the user can set various values, such as the default printer.
- The user profile are user-dependent settings (as the name implies).

- Step loops and table controls are treated identically for the recording.
- The field names of the fields listed are the same for every row.
- Each row contains a customer's bank details. The fields must be accessed using an index (this is specified as an addition to the field name, for example, KNBK-BANKN(2). The index must match the row number.
- Only the visible fields on a screen can be filled with data. If some fields are only visible after paging, the paging must be recorded as well. This means that for the recording you must execute all the steps in the online transaction.

For example *Page down* would be executed using the function code “/23”. (Paging icons)

Paging to the end of a list: “/24”.

Page up: “/22”.

Go to first page : “/21”.

- Special recording notes:
 - F1, F4 and your own F1 and F4 help (PROCESS ON HELP-REQUEST, PROCESS ON VALUE-REQUEST) are not recorded. This also applies to all commands under the *System* and *Help* menus. Values for default variants (> transaction variants) are not included.
 - Error and warning dialogs are not recorded. Only the OK code and the field contents required for further processing on the current screen are recorded.
 - Note: During recording, the data entered is written to the database

- The new “Enjoy SAP controls” developed for Release 4.6 do **not** support batch input.
- However, table controls and tabstrips do support batch input.

- With the LSMW you can create your own recordings.
- The LSMW uses the standard TA recorder here, yet evaluates the recording itself. The information from the recording is used to create batch input sessions.
- The created batch input session can be processed as usual with the batch input monitor.
- Note: As with the standard procedure it is equally important that the data is converted.

- Start recording:
 - Go into the LSMW and select the menu path *Goto ->Recordings*. An overview of all previously created recordings for your project are displayed. To start a new recording select *Create*.
 - To create a recording, you must define the name, the description and the owner.
 - Enter the name of the transaction you want to record.
 - Execute the transaction and complete it properly.
- Note: Recordings are assigned to only one project - so the name of the recording must be unique.

- In the above example the transaction *Change customer* FD02 is recorded.
- **Note:**
 - **You must make some modification to the field values for all the fields you want to capture for the data transfer. For the purposes of recording a change can simply mean overwriting one single character in a field with the same character.**
 - **Even if fields are to be filled using SET/GET parameters, the values must still be changed on the screen in order to capture them.**
 - **With online transactions screen sequences may appear that are different than in the recording.**

- The results of the recording using the LSMW are displayed in a tree structure.
- After the recording has finished, you can edit it. You can delete fields or add new ones.

- You can assign names for each recorded field. The special fields (BDC_OKCODE, BDC_CURSOR and BDC_SUBSCREEN) are not included here. You can choose the field names. When the batch input session is created, the contents of these fields are assigned to the target fields specified in the left column.
- Functions:
 - Default: Assigns to the field names of the underlying target field and its field description.
 - Reset: Deletes field name and field description.
 - Double click: Edits field name, field description and default value.
- Note: You can use field names more than once. However, the field name is only available once in the field mapping. For this reason it is not practical to use field names more than once.
- For all those fields for which you do not assign a name, the default value is used when the batch input session is created. These default values can be seen as constants. This is especially useful for check boxes (for example, MM01, view selection).

- You can later add new screen fields if you need to. Keep in mind that fields should only be added in places where they are also provided on the screen, otherwise you will get an error when the BI session is processed.
- After saving, you will get the message “The data has been saved successfully“. Now the recordings are available for the object attributes.
- If you repeat the recording, you can record a new screen sequence with new fields. Provided that the field names (e.g. KNA1-NAME1) are identical, the names and descriptions are automatically assigned.

- The recording is now available with the object attributes.
- Next, you create a project or sub-project in the LSMW and specify / process the data transfer.
- You execute the data transfer as in the standard system
 - Define the source fields and structures
 - Define the field mapping
 - Read and convert the data
 - Create and import a batch input session

- You use the transaction recorder to record transactions, and then use the transaction recording to generate a transfer program. The program allows you to transfer data using either batch input or call transaction.

- In the above example the transaction *Change customer* FD02 is recorded.

- In different transactions the module pool may change. This is the case with, for example, customer master data. The TA recorder records module pool changes automatically.

- With the TA recorder the results are also a list in a hierarchy display. However, you cannot assign your own field names as you can in the LSMW.

- You can edit the recording as needed.
- The recording can be tested and executed again. The screens are run through in the same way as in the recording.
- The recording can then be used to create:
 - Batch input sessions
 - Test data
 - Data transfer programs with batch input or call transaction
 - Function modules

- In the editor you can add or delete lines. You can also change the contents of individual lines.
- With the function *Check* you can specify whether the edited version of the recording still has the correct syntax.
- You can also export / import recordings to / from a sequential file.
- After editing save the recording changes.

- In this step, you can record additional transactions and add additional transaction runs to the recording.
- You can record the additional transactions directly after a recording or in the edit screen of a recording.

- If the editor functions in the systems do not provide all the functions you need, you can download the recording to your presentation server using the *Export* function and then edit the recording using a PC editor.
- You use the Import function to import the data back into the R/3 System. Make sure the file is still in the correct format.

- The recording can be executed directly. The data from the recording is used and the transaction is processed with call transaction.
- Call transaction provides a range of options for calling application transactions, which are discussed on the following two pages..

- Processing mode:
Here you can specify the processing mode - it is the same as the processing mode for batch input.
- Update mode:
Here you can specify the update mode. For more information see the unit call transaction.
- CATT mode:
Here you can make special settings, if you want to test the recording under CATT conditions (only necessary for CATT runs). Special functions are called in the CATT tool for each screen at the end of PBO and at the start of PAI .

- **Standard size:**
If you select this, the screens are processed in the standard size in the foreground (in foreground or display errors only). Note: the standard size is always used for background processing.
- **Further after commit:**
The first commit work in a transaction does not lead to the end of the recording.
Note: A batch input transaction is ended with the first commit work.
- **No BI mode:**
The system field SY-BINPT is not set (SY-BINPT = ' '). So transactions are run through as in online.
- **End: No BI mode:**
Only use if you choose the processing mode E (display errors only). Then the transaction is processed in the background in batch input mode and if there is an error the screens run through in the foreground.

- Not every transaction can run the same online and in batch input mode. The transaction must decide itself whether it is called online or in batch input mode.
- The system sets the system field SY-BINPT = 'X' if the batch input mode is active (this also applies to call transaction). From this system field the transaction can identify whether to run online or in batch input mode.

- You can create a batch input session directly from the recording that you can then process using the batch input monitor.

- You can create a file from the recording; the file will contain all the fields from the recording. The fields are added to the file with the length you specified in the definition.
- The field order corresponds to the order in the recording overview. Technical fields (BDC_OKCODE, BDC_CURSOR, and so on) are included.
- The data can only be stored on the application server. If you do not specify a path, the file is stored in the "work" directory of the application server.
- If the specified test file already exists, the new data record is added to this file.

- You can generate a program from the recording.
- In the following dialog box, enter a program name. In addition, you can choose the type and method for filling the maintained field contents from the recorded screens:
 - Use the values used for the recording. If you require flexible data transfer, you will have to modify this program later.
 - Parameterize the input values and read these values from a file. A data structure is created for parameterization; the data records are read from an external file into this data structure. The program assumes the file was tailored to this data structure.
- If you have decided to parameterize the input values, you should also create a test file during generation. To do this, select the checkbox and enter a name for the file.
- The attribute maintenance screen of the ABAP editor appears. Maintain the attributes and save the program.

- You have now generated a data transfer program for importing data into the R/3 System. The program can execute the data transfer using batch input or call transaction.
- If you choose option 2 and there is an error, the data record is placed into a batch input session.

- In the next dialog box you can generate a function module, a function group and short text from the recording. The function module is automatically created in the function group.

- **Import interface:**
The import interface consists of a technical part and the fields from the recording. (A parameter is created for each input field of the recorded transactions).
- **Export interface:**
Simply consists of the parameter SUBRC. The function module will return a value in the system field SY-SUBRC after calling call transaction. This return value is set to zero, if the data record has been correctly processed. (Only applies to CT)
- **If a session is created, you will be notified whether the data record could be placed into the session.**
- **Table interface:**
This is used only if the function module works with call transaction. The table contains all the system messages.

- You can use parameter CTU to specify whether the function module creates a batch input session (CTU = ' '), or updates the data record directly using call transaction (CTU = 'X')
- The other parameters are filled as for a batch input session header.
- Note: Every time the function module is called, a batch input session is created.
- If the session is to contain several transactions, proceed as follows:
Generate the transaction several times. This can be achieved by recording the transaction again. The function module interface will then contain more than one transaction. When the function module is called, several transactions can be generated for each session.

- The generated program has the structure displayed above. It uses the standard function modules for creating batch input sessions.

- The generated program consists of two parts - a generated part and the include BDCRECX1. The include is provided in the standard system.
- The include contains the definition of the required BDC table and a further seven subroutines. These subroutines execute the following functions:
 - Open and close file
 - Open and close session
 - Fill BDC table
 - Fill session
- The generated part is the main program and calls the relevant subroutines. A structure is also generated in the generated part to read the external data into the file.
- See the Appendix for more information on the BDCRECX1 program.

- The BDC table has the same structure as in a standard transfer.

- The structure of the file is automatically generated and is always defined with the name RECORD.
- The generated structure always has the same layout. Each field is defined by two generated rows. The first row is always a comment row containing the text “* data element:” and the data element of the screen field from the recording. In the next row, the field itself is defined with the name of the data element followed by a three-digit position number, which specifies the location in the structure. The length of the field is set by the length of the data element. If a field is not defined through a data element, the system uses the length of the screen field.

- The subroutine OPEN_DATASET opens the sequential file.
- The subroutine CLOSE_DATASET closes the sequential file.

- The following function modules are provided for creating batch input sessions (function group SBDC):
 - BDC_OPEN_GROUP
Creates a new session and contains general data for the whole session.
 - BDC_INSERT
Inserts all the data for a transaction into the session. The transferred internal table with the structure BDCDATA must contain all the data required for fully processing the transaction.
 - BDC_CLOSE_GROUP
Closes the session after all the data has been inserted. As soon as the session has been closed, it can be processed.
- The internal table with the structure BDCDATA can only contain batch input data for one single transaction run. So the individual transactions are inserted into the batch input session in a loop, using several BDC_INSERT calls.

- With the function module BDC_OPEN_GROUP you can open a batch input session.
- The following parameters are mandatory:
 - CLIENT (client)
 - GROUP (session name)
 - USER (user name)
 - Note: The USER must always be specified. This user name is required for processing the session in batch mode.
- The following parameters are optional:
 - HOLDDATE specifies the earliest date on which the session can be started
 - KEEP flag specifies whether session should be deleted after it has been successfully processed. The KEEP flag is ignored if the session contains errors.
 - KEEP = ' ' means delete the session after processing (default value), ;
 - KEEP = 'X' means do not delete session.
 - RECORD = 'X' allows you to create a recording instead of a BI session.
 - If QID is specified the function module returns the internal session key. This is needed to be able to differentiate sessions that have the same name.
 - With EXCEPTIONS you get various return values that can be queried with SY-SUBRC.
 - Note: In online mode the online user is used for the authorization checks regardless of the user name specified.

- The subprogram OPEN_GROUP opens the batch input session.

- With the function module BDC_INSERT you can add transaction data to the session.
- You need an internal BDC table to transfer the data.
- The following parameters are mandatory:
 - TCODE Transaction code
 - DYNPROTAB BDC table
- The following parameters are optional:
 - POST_LOCAL = 'X' activates local updating. This means that the data to be updated is kept in the application server and updated in the last step of the transaction. This does not involve the update process. (For more information see course BC414, Programming Database Updates.)
 - PRINTING = 'X' - lists processed in BI will be printed immediately.
- With EXCEPTIONS you get various return values that can be queried with SY-SUBRC.

- The subprogram BDC_TRANSACTION moves the filled BDC table either to a batch input session or uses call transaction to import the data into the system. On the selection screen, you can specify whether you want to create a session or you want to use call transaction to process the data directly. The SESSION field is then set according to your selection.

- With the function module BDC_CLOSE_GROUP you can close a batch input session.
- In EXCEPTIONS you get various return values that can be queried with SY-SUBRC.

- The subprogram CLOSE_GROUP closes the batch input session. The subprogram differentiates between a “normal” BI session (SESSION = 'X') and a session opened when call transaction encountered an error (E_GROUP_OPENED = 'X').

- The include BDCRECXX contains two subprograms BDC_DYNPRO and BDC_FIELD.
- Subprogram BDC_DYNPRO fills the BDC table with program names and screen numbers.
- Subprogram BDC_FIELD fills the BDC table with field names and values.
- Each subprogram creates an entry in the BDC table.

- The BDC table is filled with data from the recording. The subroutines BDC_DYNPRO and BDC_FIELD are used for this.
- The subroutines are called in the required structure or sequence.

- In contrast to batch input, you can use call transaction to directly pass data to the dialog interface, without using a batch input session. You use an internal table (BDC table, same structure and layout as batch input) to temporarily store your screen data. Then you call the desired transaction in your program.
- Note: On average, the Call Transaction method is 1 to 3 times faster than using Batch Input.

- An additional system roll area is added for the called transaction. The system uses this area for processing the individual screens in the transaction; the data from the table is added here. After the transaction is processed, the second roll area is released; the system continues processing in the first roll area.

Notes:

In contrast to batch input, there is no error logging (error sessions) here. For processing the called transaction, the system uses the authorizations of the user that called the transaction. The BDC table can only hold data for one transaction run at a time. Before another transaction is called and its data can be added, you must execute a REFRESH for the BDC table.

- With the parameter `MODE` you can select a specific display mode for processing the transaction. The following three modes are available:
- `A` (Display all)
If you execute the program, all the screens and all their data are displayed. This is the default value for the `MODE` in `CALL TRANSACTION`.
- `N` (Display nothing)
The screens are not displayed at all, regardless of whether any of them contained errors. As soon as the transaction has been fully processed, control is returned to your program. (The value of the parameter `UPDATE` determines whether database updates are executed or not).
- `E` (Display errors only)
As soon as an error occurs in a screen, the transaction is switched to display mode so that you can correct the errors.
- These display modes are the same as those in the batch input session processing.

- With the parameter UPDATE you can control how database updates arising from a transaction are carried out. There are three options:
- A (**asynchronous updating**) The called transaction does not wait until the database has been updated, it simply forward the updates to the SAP update service. This usually speeds up the CT program. This processing mode is NOT recommended for large data sets as the called transaction does not receive a completion message from the update module.
The calling CT program therefore cannot tell if the transaction has been successfully completed. You should use the updating administration function (transaction SM13) to determine whether updating was terminated early. The error analysis /correction functions for are less helpful than with synchronous updating.
- S (**synchronous updating**) With synchronous updating the called transaction waits until all the updates have been completed. So processing is slower than with synchronous updating. The called transaction can however report any updating errors to the program, which makes error analysis/correction easier.
- L (**local updating**) with local updating the database is not updated in a separate process , it is updated in the caller's process (see application help documentation for the ABAP keyword SET UPDATE TASK LOCAL)

- **MESSAGES** specifies that all system messages that are output during the execution of a **CALL TRANSACTION** are written to the internal table <MESSTAB>. The internal table must have the structure **BDCMSGCOLL**.
- Unlike the classical batch input processing with sessions **CALL TRANSACTION** does not offer any special processing procedures for transactions containing errors. There are no restart functions for transactions that contain errors or that cause updating terminations.

- The addition `OPTIONS` enables the call transaction statement to be controlled through a structure. The structure must have the dictionary type `CTU_PARAMS`.
- The addition `OPTIONS FROM` must **not** be combined either with the addition `MODE` or with `UPDATE`. More information on the next slide.

- The components are defined as follows:
 - The components DEFSIZE , RACOMMIT, NOBINPT, and NOBIEND can have the following values: 'X' = Yes or ' ' = No
- If the addition OPTIONS FROM is not specified, the following settings apply to the control parameters:
 - DISMODE from addition MODE
 - UPDMODE from addition UPDATE
 - CATTMODE CATT not active
 - DEFSIZE Do not not use default windows size
 - RACOMMIT Successful exit at COMMIT WORK
 - NOBINPT Batch input session active (SY-BINPT = X)
 - NOBIEND Batch input session still active after end of BDC data

- The return code is used to specify whether the called transaction was processed successfully. In addition, several system fields are filled: The number, ID, type, and variables of the dialog message output by the called transaction. You need this information to output the message text.

- If you have selected call transaction on the selection screen, the generated program runs as follows:
 - The online transaction is called with call transaction and passed to table BDCDATA. All the messages arising during the call transaction are put in the internal table MESSTAB.
 - MESSTAB is output as a list (see next slide).
 - If you have specified an error session (E_GROUP <> SPACE) on the selection screen, and the call transaction comes across an error (L_SUBRC <> 0), the error session is opened. The error session is however only opened once. Then the data record is placed in the session.
- All the other data records that have errors with call transaction are placed in the same session.

- You can integrate the generated program into the DX-WB. The structure of the file must be created in the ABAP Dictionary. The tables SXDA0, SXDA1, SXDA2 and SXDA3 will be filled.

- The recorder can be integrated as of Release 4.5 and it is provided in the DX-WB Release 4.5.
- Got to the DX-WB Release 4.5. On the initial screen you can integrate programs by choosing *Goto*
-> *Int. customer data transfer object*.

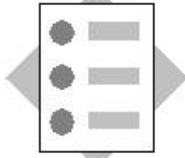
- For Release 4.5, transfer objects were not organized by BOR objects but by a DX number (0001-9999). The number range 9000-9999 is reserved for customers.
- To create a connection to the new DX -WB, you must assign the program to a BOR object.
- Create a short text for the DX object. This text will appear in the method selection list.
- Specify the program to be included.
- Specify a name for the structure to be created in the ABAP Dictionary.
- **Note: The generated program appears under program type BINP and not under program type REPO.**

- If you included the object in the Release 4.5 workbench, it is automatically available in the Release 4.6 workbench.
- All the workbench functions are available for your object except for *Create files with transfer data*.

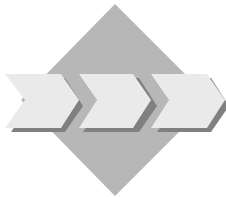


Unit: Transaction Recorder

Topic: Recording function in the LSMW



- Create a recording using the LSMW
- Generate a batch input session



Change the debtor address data in the R/3 System



Name of the recording: DEBI_##

Name of the file: BC420-##-BIFILE

- 1 Create the recording DEBI_## using the LSMW. Record the transaction FD02 (Change debtor).
 - 1-1 Before you create the recording, first test the transaction online!
For the recording use debtor Z-##-10001. On the initial screen of the FD02, maintain the following fields only:
Debtor: Debtor Z-##-10001
Company code: 0001
Address data: select
Change the following fields:
Name, city, street, postal code and telephone number!

Note: The data record must be saved while recording!
 - 1-2 Maintain the names and the descriptions of the fields. Use the same technical names as shown in the table below (automatic field mapping).

2 Create the subproject DEBI-## and the new object TA_## in your project BC420-##.

Maintain the object attributes and select the item batch input recording. Select your recording DEBI_##.

Create the source structure address.

Create the source fields according to the table below. Or copy them from project BC420-S_EN, subproject DEBI-00, object TA-00.

2-1 File setup: The data from the legacy system are displayed in the following table:

Field name	Field type	Field length	Description
NO	C	16	Debtor number
NAME	C	35	Name
STREET	C	35	Street
CITY	C	35	City/Place of residence
ZIP	C	10	Postal code
TEL	C	16	Telephone number

2-2 Maintain the field mapping. Use the automatic field mapping (via menu item: Extras)

2-3 Specify the files. Only maintain the file name of the source file.

Name of the file BC420-##-BIFILE

Codepage: 1100

2-4 Import the data and convert them. Display the data in each case!

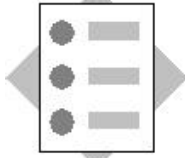
2-5 Create a batch input session

2-6 Run the batch input session.

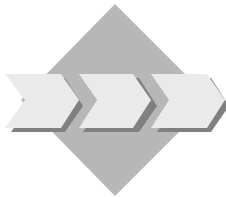


Unit: Transaction Recorder

Topic: Recording with the TA Recorder



- Generate a recording using the TA recorder.
- Test the recording.



Change the address data of debtors in the R/3 system



Name recording: TA_DEBI_##

Name of the file: BC420-##-BIFILE

- 1 Create the recording TA_DEBI_## using the transaction recorder. Record the transaction FD02 (change debtor).
 - 1-1 Use debtor Z-##-10001 for the recording. On the FD02 initial screen, only maintain the following fields:

Debtor: Debtor Z-##-10001

Company Code: 0001

Address data: select

Record the following fields:

Name. Street, city, area code and telephone number!

Note: The data record must be saved when recording!

Save the recording now.
- 2 Switch to the recordings overview.
 - 2-1 Test the recording with the processing mode „A“ (display all).
 - 2-2 Test the recording with the processing mode „N“ (no display).
 - 2-3 If the recording has been tested and no error has occurred, generate a BI session and run it using the BI monitor. If an error should occur, generate the recording again!
- 3 From the recording, generate the program ZBC420_DEBI_## in your development

- Read field contents from file
 - Create test data in the file BC420_debi_test_##.txt
 - 3-1 Save this program and execute it. Generate one BI session each and process it. Carry out the same steps again using the call transaction.
- 4 The generated program should now be changed so that file BC420-##-BIFILE can be read and processed.
 - 4-1 Change the default value of the file name (parameter dataset) in BC420-##-BIFILE.
 - 4-2 Change the structure definition. Delete the company code field and the address data checkbox from the structure
 - 4-3 Change the sub-routine call `perform bdc_fi el d` for the company code (' **RF02D-BUKRS** ') so that the value 0001 is transferred.
 - 4-4 Change the sub-routine call `perform bdc_fi el d` for the address data checkbox (' **RF02D-D0110** ') so that the value X is transferred.
 - 4-5 Start the program again and read the data from the file BC420-##-BIFILE with debtor data. Generate a BI session with it and run it.

***INCLUDE BDCRECX1.

SELECTION-SCREEN BEGIN OF LINE.

PARAMETERS SESSION RADIOBUTTON GROUP CTU. "create session
SELECTION-SCREEN COMMENT 3(20) TEXT-S07 FOR FIELD SESSION.
selection-screen position 45.

PARAMETERS CTU RADIOBUTTON GROUP CTU. "call transaction
SELECTION-SCREEN COMMENT 48(20) TEXT-S08 FOR FIELD CTU.

SELECTION-SCREEN END OF LINE.

SELECTION-SCREEN BEGIN OF LINE.

SELECTION-SCREEN COMMENT 3(20) TEXT-S01 FOR FIELD GROUP.
selection-screen position 25.

PARAMETERS GROUP(12). "group name of session

SELECTION-SCREEN COMMENT 48(20) TEXT-S05 FOR FIELD CTUMODE.
selection-screen position 70.

PARAMETERS CTUMODE LIKE CTU_PARAMS-DISMODE DEFAULT 'N'.

"A: show all dynpros

"E: show dynpro on error only

"N: do not display dynpro

SELECTION-SCREEN END OF LINE.

SELECTION-SCREEN BEGIN OF LINE.

SELECTION-SCREEN COMMENT 3(20) TEXT-S02 FOR FIELD USER.
selection-screen position 25.

PARAMETERS: USER(12) DEFAULT SY-UNAME. "user for session in batch

SELECTION-SCREEN COMMENT 48(20) TEXT-S06 FOR FIELD CUPDATE.
selection-screen position 70.

PARAMETERS CUPDATE LIKE CTU_PARAMS-UPDMODE DEFAULT 'L'.

"S: synchronously

"A: asynchronously

"L: local

SELECTION-SCREEN END OF LINE.

SELECTION-SCREEN BEGIN OF LINE.

SELECTION-SCREEN COMMENT 3(20) TEXT-S03 FOR FIELD KEEP.
selection-screen position 25.

PARAMETERS: KEEP AS CHECKBOX. "' ' = delete session if finished
"'X' = keep session if finished

SELECTION-SCREEN COMMENT 48(20) TEXT-S09 FOR FIELD E_GROUP.
selection-screen position 70.

parameters E_GROUP(12). "group name of error-session

SELECTION-SCREEN END OF LINE.

SELECTION-SCREEN BEGIN OF LINE.

SELECTION-SCREEN COMMENT 3(20) TEXT-S04 FOR FIELD HOLDDATE.

selection-screen position 25.

PARAMETERS: HOLDDATE LIKE SY-DATUM.

SELECTION-SCREEN COMMENT 51(17) TEXT-S02 FOR FIELD E_USER.

selection-screen position 70.

PARAMETERS: E_USER(12) DEFAULT SY-UNAME. "user for error-session

SELECTION-SCREEN END OF LINE.

SELECTION-SCREEN BEGIN OF LINE.

SELECTION-SCREEN COMMENT 51(17) TEXT-S03 FOR FIELD E_KEEP.

selection-screen position 70.

PARAMETERS: E_KEEP AS CHECKBOX. " ' ' = delete session if finished

" 'X' = keep session if finished

SELECTION-SCREEN END OF LINE.

SELECTION-SCREEN BEGIN OF LINE.

SELECTION-SCREEN COMMENT 51(17) TEXT-S04 FOR FIELD E_HDATE.

selection-screen position 70.

PARAMETERS: E_HDATE LIKE SY-DATUM.

SELECTION-SCREEN END OF LINE.

SELECTION-SCREEN SKIP.

SELECTION-SCREEN BEGIN OF LINE.

SELECTION-SCREEN COMMENT 1(33) TEXT-S10 FOR FIELD NODATA.

PARAMETERS: NODATA DEFAULT '/' LOWER CASE. "nodata

SELECTION-SCREEN END OF LINE.

* Batchinputdata of single transaction

DATA: BDCDATA LIKE BDCDATA OCCURS 0 WITH HEADER LINE.

* messages of call transaction

DATA: MESSTAB LIKE BDCMSGCOLL OCCURS 0 WITH HEADER LINE.

* error session opened (' ' or 'X')

DATA: E_GROUP_OPENED.

* message texts

TABLES: T100.

```

*-----*
*   at selection screen                                     *
*-----*

AT SELECTION-SCREEN.
* group and user must be filled for create session
  IF SESSION = 'X' AND
    GROUP = SPACE OR USER = SPACE.
    MESSAGE E613(MS).
  ENDIF.

*-----*
*   open dataset                                           *
*-----*

FORM OPEN_DATASET USING P_DATASET.
  OPEN DATASET P_DATASET IN TEXT MODE.
  IF SY-SUBRC <> 0.
    WRITE: / TEXT-E00, SY-SUBRC.
    STOP.
  ENDIF.
ENDFORM.

*-----*
*   close dataset                                          *
*-----*

FORM CLOSE_DATASET USING P_DATASET.
  CLOSE DATASET P_DATASET.
ENDFORM.

*-----*
*   create batchinput session                             *
*   (not for call transaction using...)                   *
*-----*

FORM OPEN_GROUP.
  IF SESSION = 'X'.
    SKIP.
    WRITE: /(20) 'Create group'(I01), GROUP.
    SKIP.
  *   open batchinput group
    CALL FUNCTION 'BDC_OPEN_GROUP'
      EXPORTING CLIENT   = SY-MANDT
              GROUP     = GROUP
              USER      = USER

```

```

                KEEP      = KEEP
                HOLDDATE = HOLDDATE.
WRITE: / (30) 'BDC_OPEN_GROUP' (I02),
        (12) 'returncode:' (I05),
        SY-SUBRC.

ENDIF.
ENDFORM.

*-----*
*   end batchinput session                               *
*   (call transaction using...: error session)          *
*-----*

FORM CLOSE_GROUP.
  IF SESSION = 'X'.
*   close batchinput group
  CALL FUNCTION 'BDC_CLOSE_GROUP'.
  WRITE: / (30) 'BDC_CLOSE_GROUP' (I04),
        (12) 'returncode:' (I05),
        SY-SUBRC.

ELSE.
  IF E_GROUP_OPENED = 'X'.
    CALL FUNCTION 'BDC_CLOSE_GROUP'.
    WRITE: /.
    WRITE: / (30) 'Fehlermappe wurde erzeugt' (I06).
  ENDIF.
ENDIF.
ENDFORM.

*-----*
*           Start new transaction according to parameters *
*-----*

FORM BDC_TRANSACTION USING TCODE.
  DATA: L_MSTRING(480).
  DATA: L_SUBRC LIKE SY-SUBRC.
* batch input session
  IF SESSION = 'X'.
    CALL FUNCTION 'BDC_INSERT'
      EXPORTING TCODE      = TCODE
      TABLES   DYNPROTAB = BDCDATA.
    WRITE: / 'BDC_INSERT' (I03),
          TCODE,
          'returncode:' (I05),

```

```

        SY-SUBRC,
        'RECORD:',
        SY-INDEX.
* call transaction using
ELSE.
    REFRESH MESSTAB.
    CALL TRANSACTION TCODE USING BDCDATA
        MODE    CTUMODE
        UPDATE  CUPDATE
        MESSAGES INTO MESSTAB.
L_SUBRC = SY-SUBRC.
WRITE: / 'CALL_TRANSACTION',
        TCODE,
        'returncode:'(I05),
        L_SUBRC,
        'RECORD:',
        SY-INDEX.
LOOP AT MESSTAB.
    SELECT SINGLE * FROM T100 WHERE SPRSL = MESSTAB-MSGSPRA
        AND   ARGBG = MESSTAB-MSGID
        AND   MSGNR = MESSTAB-MSGNR.
IF SY-SUBRC = 0.
    L_MSTRING = T100-TEXT.
    IF L_MSTRING CS '&1'.
        REPLACE '&1' WITH MESSTAB-MSGV1 INTO L_MSTRING.
        REPLACE '&2' WITH MESSTAB-MSGV2 INTO L_MSTRING.
        REPLACE '&3' WITH MESSTAB-MSGV3 INTO L_MSTRING.
        REPLACE '&4' WITH MESSTAB-MSGV4 INTO L_MSTRING.
    ELSE.
        REPLACE '&' WITH MESSTAB-MSGV1 INTO L_MSTRING.
        REPLACE '&' WITH MESSTAB-MSGV2 INTO L_MSTRING.
        REPLACE '&' WITH MESSTAB-MSGV3 INTO L_MSTRING.
        REPLACE '&' WITH MESSTAB-MSGV4 INTO L_MSTRING.
    ENDIF.
    CONDENSE L_MSTRING.
    WRITE: / MESSTAB-MSGTYP, L_MSTRING(250).
ELSE.
    WRITE: / MESSTAB.
ENDIF.
ENDLOOP.
SKIP.
** Erzeugen fehlermappe *****

```

```

IF L_SUBRC <> 0 AND E_GROUP <> SPACE.
  IF E_GROUP_OPENED = ' '.
    CALL FUNCTION 'BDC_OPEN_GROUP'
      EXPORTING CLIENT    = SY-MANDT
              GROUP     = E_GROUP
              USER      = E_USER
              KEEP      = E_KEEP
              HOLDDATE  = E_HDATE.
    E_GROUP_OPENED = 'X'.
  ENDIF.
  CALL FUNCTION 'BDC_INSERT'
    EXPORTING TCODE      = TCODE
    TABLES   DYNPROTAB = BDCDATA.
  ENDIF.
ENDIF.
REFRESH BDCDATA.
ENDFORM.

```

```

*-----*
*           Start new screen                               *
*-----*

```

```

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

```

```

*-----*
*           Insert field                                   *
*-----*

```

```

FORM BDC_FIELD USING FNAM FVAL.
  IF FVAL <> NODATA.
    CLEAR BDCDATA.
    BDCDATA-FNAM = FNAM.
    BDCDATA-FVAL = FVAL.
    APPEND BDCDATA.
  ENDIF.
ENDFORM.

```

Contents:

- **Transferring data using interactive lists**
- **Using variable external data format**
- **SAP - LUW**
- **Changing RFBIDE00 to call transaction**
- **Tips & Tricks**

- The demo program SAPBC420_SPTD_INTERACTIV_LIST creates a list from which you can select details by choosing checkboxes.

- When this transaction is recorded using the transaction recorder, you can see that the check boxes in the recording list are not checked. These important entries must be **added manually**.
- To do this, you must specify the position in the list using X/Y coordinates (X is the row, and Y is the column).
Important to note: The first column in the list has the value 2.
- **The selected row does not always match the row position, as the headers must also be included in the total number of rows.**
- The fourth check box selected in the example is actually the sixth row as the header includes two rows. The value for the BDC_CURSOR cursor field is set to '07/02', as this is the last position on the last.
The fields to select are therefore row 6/ column 2 and row 7/ column 2.

- The slide illustrates the changes made in the recording editor.

- In the example, the external data is in a variable format. All customer information has the record type “A” except for the bank details. The first byte in every record indicates the record type.
- The bank details for a customer are listed after the A records and are marked as record type “B”.

- The slide shows an example of how to read the variable external data format.
- If the data is binary data, the system first reads the record indicator into the variable *rectype*.
- The form routine *fill_tab* is used to interpret and read the rest of data according to the record indicator (A or B). Processing can then begin.

- From a business point-of-view, an SAP logical unit of work (SAP-LUW) consists of an SAP transaction a user executes online (first part of LUW) and the corresponding update (second part of LUW). In online processing, the user can proceed with the next SAP transaction after saving, usually at the end of the transaction processing (online part). The user therefore starts an additional SAP-LUW while the update from the first SAP-LUW was running. This is asynchronous transaction processing.
- In online processing, the user can proceed with the next SAP transaction after saving, usually at the end of the transaction processing (online part). The user therefore starts an additional SAP-LUW while the update from the first SAP-LUW was running. This is called asynchronous transaction processing.
- **The processing of batch input sessions, however, is synchronous. This means SAP-LUW 2 is not started until the update for SAP-LUW 1 is completed.**

- Processing mode "A" for call transaction:
After the first part of the SAP-LUW is completed and the changes are marked, the next SAP-LUW can be started immediately. This means that external data imported using call transaction could be imported partially in parallel (overlapping import), if the update takes longer than the calling of the next record to be processed.
- We do not recommend the use of asynchronous processing with CT; if you do use it, test it thoroughly beforehand.
SAP Notes on updating / locks:
 - Lock table 17267, 13907, 97760
 - Update / repeat update 70085

- When the parameter XCALL is selected, processing is through 'Call Transaction .. Using ..'. The transactions are not stored in a batch input session, instead, processing begins immediately. If processing with 'Call Transaction .. Using ..' was not successful, the cause of the error is logged, and the transaction is then saved in a batch input session.
- You can use parameter ANZ_MODE to control the display mode (see CT method).
- The parameter UPDATE specifies the update mode (see CT method).

- Program RFBIDE00 for customer data transfer includes comments for help with modifying the program for use with call transaction.
- Comment out the three variables after the DATA statement. Set the three variables after the PARAMETERS statement to active (remove comments).

- SAPNet allows you to create your own problem messages for SAP support, thus speeding up and optimizing processing of customer messages.
- You can perform a free search of the SAP Note database for answers to questions you may have.

RSBDCREO is the most important of the three utility programs listed. You use this program to reorganize batch input sessions and their logs.

- Database rollback segments are buffer areas that store the “before image” of the database during a database logical unit of work (LUW) (DB - LUW = database processing step). The “before image” is the change information needed to restore the database to a consistent state, if an error occurs during this small processing step.
- The call of BDC_INSERT to fill the batch input session causes database changes that fill the rollback segments. To restrict the growth of these segments, we recommend you trigger a database commit at regular intervals (every 100 or 1000 loops). To do this, use the ABAP command COMMIT WORK, which resets the rollback segments.

- Logs:
The batch input logs are stored in the directory DIR_GLOBAL (../global) on the application server. You should use the ABAP program RSBDCREO to reorganize these files.
- Tablespace size:
Depending on the size of the external data to be transferred to the R/3 System, the batch input session may exceed the size of the tablespace. If this happens, the database administrator must increase the size of tablespace PSA PSTABD.

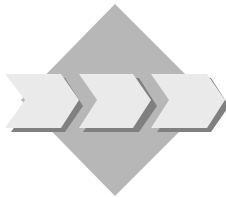


Unit: Special Methods

Topic: Interactive Lists and Batch Input



- Data transfer with interactive lists



Select the check boxes of an interactive list using the the batch input/call transaction method. The transaction recorder first records a program and then edits the recording.



Recording: **ILIST_##**

Transaction to be recorded: **BC420**

- 1 Recording of interactive lists.
 - 1-1 The program SAPBC420_SPTD_LIST_CHECKBOXES displays customer master data via an interactive list. Call this program, flag some of the checkboxes and then click on the magnifying glass icon → Detailed display ! (The detailed display shows the debtor addresses)
 - 1-2 Record this program (transaction code: BC420).
 - 1-3 Execute the recording for test purposes! Have the processing of the list and the setting of the check boxes been executed correctly?

 - 1-4 Edit and correct the recording, so that the check boxes can be set. You will find the relevant information in the training course slides!



Chapter: Special techniques

Solution to 1.3

1 Recording interactive lists

- 1-3 The list is not processed correctly. The checkboxes are not set in the recording. If for example the fields seven and eight are checked, enter the following to the recording below the module pool line SAPMSSY0, 120, X:

Field name	Field value
"09/02"	, "X" and
"10/02"	, "X"

This marks the checkboxes seven and eight.

The two lines above the first checkbox (list header and underline) are added to the line numbering in the output list, so the line numbers nine and ten in the example mark the checkboxes seven and eight (the "02" is the column number).

Contents:

- **Scheduling jobs**
- **Using the job overview**
- **Using parallel BI processing through RSBDCSUB**

- R/3 background processing is used to automate routine tasks and optimize the use of the R/3 resources within a company. You or your users schedule programs to be run by the R/3 System and specify which programs to execute and when.
- The R/3 System provides comprehensive support for background processing including a number of methods for scheduling and maintaining jobs. You can execute both internal R/3 and external programs. To simplify scheduling, you can execute related internal and external jobs as “job steps” within one background processing job. Using job steps, you can define a complex background job that includes several processing steps. Job scheduling includes all process activities required to execute the task.
- In addition, there are various job administration and diagnostic tools, including a graphical monitor. There is a powerful, easy-to-use programming interface you can use to program your own applications for background processing, if needed; you can use the job wizard to schedule your jobs. The wizard automates the basic definition of the background processing job and guides new users through the entire process.

- To create a background processing job, proceed as follows:
 - 1) If the program to be scheduled contains a selection screen, create a variant.
 - 2) In the job definition, you must specify the job name and priority.
 - Optional: Target server
 - 3) The step list contains all programs to be sequentially processed within a job.
 - 4) You must set the start time for the job.
 - 5) When the start time is reached, the job is processed in a free background work process.
 - 6) You can then display the job logs.

- In the job definition, you specify a job name and job class (A = highest, to C = lowest priority).
- To ensure automatic workload balancing, leave the *Target server* field empty. The system then executes the job on a server with free background work processes.
- Under *Steps* you define the individual job steps (individual programs within the job).
- Under *Start conditions*, you specify the start date and time.

- A step can be an internal ABAP program. If the program contains a selection screen, you must create a variant, which is a saved set of selection screen values for this program.

- After you save the steps, the step list appears. To add additional steps, choose *Create*.
- After you define and *Save* the step list and go *Back* to the initial screen, you can specify the start date and time for the job.

- One scheduling method is to specify a date and time.
- Not only can you specify a start date and time, you can also specify an “expiration date”.
If the job has not been processed by this date and time (because there is no free background work process), it will never be processed. The job status is then set to *Canceled*.
- You can also specify a *Periodic* start time.

- You can use the job overview to select and display jobs according to specific criteria.
- For example, you could choose to display all jobs that have status *Ready*, but that have not yet started.
- If you want to display scheduled jobs that depend on an event, you must enter at least “*” in the *Start after event* field.

- From the job overview, you can go to the Spool transaction to display the list output of the individual job steps.
- To view the job logs, choose *Job log*.
- Job status:
 - Scheduled: Job is defined but no start time has been specified.
 - Released: Start time has been specified
 - Ready: Execution time has been reached, job ready for execution
 - Active: Job running, see transaction SM50
 - Finished: Job completed without errors
 - Canceled: Job ended; Errors occurred.

- You can process BI sessions in parallel in the background using the program RSBDCSUB. If several BI sessions were generated from the external data, RSBDCSUB schedules a separate job for each BI session.
- On the selection screen of RSBDCSUB, you specify the names of the BI sessions you want to process (generic entry possible: "bc420*").

Using program RSBDCSUB, you can schedule session processing in the background.

Proceed as follows:

- 1) Test RSBDCSUB to learn about the input parameters on the selection screen and the program's functions.
- 2) Create a variant of RSBDCSUB using the following selection criteria:
 - Session name (generic entries also possible, such as "bc420*")
 - Created on
 - Session status
 - It's a good idea to leave the *Target server* field empty (for automatic load balancing).
 - Extended log
- 3) Create a background job for program RSBDCSUB using the variant you created.

- The external data conversion and import of the data into the R/3 System can be automatically scheduled through background processing.
- This requires two program steps:
 - 1) In Step 1, the BI program is called that reads the external data and converts the data. BI sessions are then created.
 - 2) In steps 2, RSBDCSUB is called and transfers the generated BI sessions to background processing.

- The job overview contains the results of the individual job runs (one job per session).
- The results of the session processing can also be viewed in the BI monitor.

- The "DOWNLOAD" and "UPLOAD" function modules are designed for online operation and **cannot be executed in the background.**

- 2 Define and schedule a job to generate and run batch input sessions. Define one job “BC420_##_Job” that performs the following tasks in two steps:
- a) Run a program to generate one or more batch input sessions. Before you start, generate a variant for the scheduled batch input program.
 - b) Call the RSBDCSUB to schedule the batch input session that has been created in the previous step. You must also first create a variant for RSBDCSUB.

Steps:

- 2-1 Create the variant VAR_## for the program RFBIBL00. Maintain the
 - File name = (File name see outbound file respectively converted data in the LSMW object CT-##).
 - Type of data transfer is batch input
- 2-2 Create a variant for the RSBDCSUB. This variant should process the batch input sessions CT-##. (This session name is from the LSMW outbound file and corresponds to the object name).
- 2-3 Schedule a job consisting of two steps: The first step contains your batch input program RFBIBL00, the second step contains RSBDCSUB. The variants you defined before are necessary for both steps.
- 2-4 In order to run BI sessions, follow the flow of the job and check the job log for each job that has been created by the RSBDCSUB.

- 3 Optional: Generating several BI sessions by changing an LSMW setting and scheduling the LSMW main program.

- 3-1 Display all objects for your project. Select the object CT-## and copy it to the new object BI-##.
- 3-2 Make the following changes in the END_OF_RECORD of structure BGR00. The field `g_cnt_transactions_group` determines the number of records according to which the next BI session is generated. Change this value to 2.

```
at_first_transfer_record.  
if g_cnt_transactions_group = 2.  
g_cnt_transactions_group = 0.  
transfer_record.  
endif.
```

- 3-3 In the object attributes set the data transfer to *periodic*. Then you get the main program `/SAPDMC/SAP_LSMW_INTERFACE`, which will perform the steps of the LSMW automatically.

3-4 Create the variant Var-## for this main program.

Settings:

Project: BC420-##

Sub-project: DOCU-##

Object: BI-##

Set the parameters of some batch input/direct input programs
“BI, DI, Call Transaction” to “B”.

3-5 Schedule a job that consists of one step and that contains the main program with the variant VAR-##. Start the job.

3-6 Follow the the job flow and check the job log.



Chapter: Background processing Solution to 3.2

3 Optional: Create several BI sessions by changing an LSMW setting and scheduling the LSMW main program.

3-2 Go to Extras -> Display variant and set the "Technical field" and "Processing events" checkboxes to change a technical LSMW field mapping setting. "Coding" and "Global data definitions" should also be selected.

The processing event "END OF RECORD" is below the structure BGR00 and determines how many transactions a new BI session processes with a counter.

Change the test "if g_cnt_transactions_group = 5000" to "if g_cnt_transactions_group = 3" to create a new BI session after reading three records.