



Guideline

Data Migration of FI-CA-Based Industry Solutions

IS Migration Workbench

Document Version 1.03 - January 2011

Copyright

© 2011 by SAP AG.

All rights reserved. SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects S.A. in the United States and in other countries. Business Objects is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.




Document History

Version	Date	Comment
1.01	01/05/2009	1 st version of document published
1.02	01/08/2009	Migration scenario <i>Payment Scheme</i> Migration of meter readings Updated FAQ and code snippets
1.03	21/01/2011	Updated FAQ and code snippets Contract Accounting: Migration of taxes of open item Contract Accounting: Migration of rounding balances SAP for Utilities: Migration of meter reading SAP for Utilities: Migration of AMI devices SAP for Utilities: Migration of formula profiles SAP for Utilities: Update of BP address list during move-in SAP for Utilities: Update of search index for BP address

Typographic Conventions

Type Style	Description
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbutton labels, menu names, menu paths, and menu options. Cross-references to other documentation
<i>Example text</i>	Emphasized words or phrases in body text, graphic titles, and table titles
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
Example text	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
Example text	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
<Example text>	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.

Icons

Icon	Meaning
	Caution
	Example
	Note

Motivation

The guideline **Data Migration of FI-CA-Based Industry Solutions – IS Migration Workbench** is aimed at customers, SAP consultants, and partners who are responsible for configuring and operating the IS Migration Workbench (ISMW). This document supplements the following:

- SAP course *IUTW90*
- *User Handbook* in the IS Migration Workbench
- Cookbook *IS Migration Performance*
- Further documentation is available on **SAP Help Portal** at <http://help.sap.com> and in the **SAP Service Marketplace** at <http://service.sap.com/> and <http://service.sap.com/utilities>.
- *Release notes for ECC 6.0*

The purpose of the document is to provide guidelines and share experiences made with the IS Migration Workbench to enable more efficient usage of the tool. Functions available in the IS Migration Workbench are explained only if necessary to understand the solution of a problem. The document highlights functions that are not often used.

The document has the following structure:

Chapter 1 introduces the IS Migration Workbench.

Chapter 2 describes the most important functions of the IS Migration Workbench in detail.

Chapter 3 introduces the available migration objects and explains step-by-step how to generate your own migration objects.

Chapter 4 gives an insight into the generation of the load report. It explains step-by-step how to enhance load programs by including your own code in the load report.

Chapter 5 shows how to migrate data into custom tables.

Chapter 6 describes various important data migration strategies that allow the understanding, designing and planning of the data migration process, based on the business processes to be migrated.

Chapter 7 answers project-specific questions. To do this, we analyzed many customer messages to present the most common problems, questions, and answers.

Chapter 8 gives an overview of the problems of data migration from one SAP system to another.

Chapter 9 shows examples of code snippets which you can use as a basis for your own implementation.

Chapter 10 explains step-by-step how to get started with the IS Migration Workbench from the initial configuration to the data import of some sample data objects.

Author

Friedrich Keller, SAP AG

Feedback

We would appreciate your feedback on this document. Please send your comments to <mailto:friedrich.keller@sap.com>.

Table of Contents

1 IS Migration Workbench.....	10
1.1 Introduction to the IS Migration Workbench.....	10
1.2 Proven Concepts.....	10
1.3 General Description	11
2 Functions in Detail.....	14
2.1 Load Program	14
2.2 Migration Company	14
2.3 User Parameters	15
2.4 Field Rules	16
2.4.1 Field Rule Initial Value	17
2.4.2 Field Rule <i>Fixed Value</i>	17
2.4.3 Field Rule <i>Transfer</i>	19
2.4.4 Field Rule <i>ABAP Rule</i>	19
2.4.5 Field Rule <i>via KSM</i>	20
2.4.6 Field Rule <i>Conversion</i>	20
2.4.7 Field Rule Customer Field	22
2.5 Key and Status Management.....	23
2.5.1 General Description.....	23
2.5.2 Implementation.....	23
2.5.3 Use of the KSM in Field Rules	24
2.5.4 Planning for Unique Oldkeys	25
2.6 Data Import	25
2.6.1 Import File	25
2.6.2 Import File Transformation.....	27
2.6.3 Import File Editor	27
2.6.4 Data Import.....	29
2.6.5 Migration Statistics	30
2.7 Distributed Import	34
2.7.1 General Description.....	34
2.7.2 Implementation	34
2.7.3 Group Import	37
2.7.4 Alternative Job Distribution	38
2.7.5 User-Exit in Master Job	39
2.8 Migration Lock Mechanism.....	40
2.9 Authorization	43
2.10 Change Requests	45
2.10.1 Transport with Change Requests.....	45
2.10.2 Transport of Migration Objects in Text Files.....	46
3 Migration Object	47
3.1 Definition.....	47
3.2 Migration Class.....	49
3.3 Pre-Configured Migration Objects.....	49
3.3.1 Migration Objects for all Solutions.....	50
3.3.2 Migration Objects for Regional Structure	50
3.3.3 Migration Objects for Business Master Data	51

3.3.4 Migration Objects for Financial Data	52
3.3.5 Migration Objects for Miscellaneous Data.....	54
3.3.6 Migration Objects Specific to SAP for Utilities	54
3.3.7 Migration Objects Specific to SAP for Waste and Recycling	62
3.3.8 Migration Objects for Budget-Billing Plans	63
3.3.9 Migration Object for Plant Maintenance (PM)	64
3.3.10 Migration Objects specific to SAP for Insurance	64
3.3.11 Migration Objects specific to SAP for Public Sector	65
3.4 Generation of Own Migration Objects.....	66
3.4.1 Generation of a BDC Migration Object.....	66
3.4.2 Generation of a BAPI Migration Object	69
3.5 Hyper Object.....	72
3.5.1 Motivation.....	72
3.5.2 Restrictions.....	72
3.5.3 Creation Procedure (Hyper Object).....	72
3.5.4 Creation Procedure (Reference to Migration Object)	74
3.5.5 Generation of the Load Report	76
3.6 Migration Control Parameter	78
3.6.1 General Control Parameters.....	78
3.6.2 Control Parameters Specific to SAP for Utilities.....	78
3.6.3 Configuring Control Parameters	81
3.6.4 Function Module ISU_MIG_STATUS.....	81
4 Enhancing a Load Program	83
4.1 Motivation	83
4.2 Layout of the Load Program.....	83
4.3 Custom Code in a Load Program	84
4.3.1 Code on Field level.....	85
4.3.2 Code on Structure Level.....	86
4.3.3 Code on Report Level.....	87
4.3.4 Error Handling in Custom Code	89
4.4 Customer-Defined or Additional Fields.....	90
4.4.1 Additional Fields in the IS Migration Workbench.....	90
4.4.2 Enhancement Fields in the Solution.....	92
4.4.3 Enhancement Fields for Technical Master Data (SAP for Utilities).....	93
4.4.4 Enhancement Fields for a Point of Delivery (SAP for Utilities)	93
4.4.5 Enhancement Fields for Business Partners	94
4.4.6 Enhancement Fields for Contract Accounts.....	95
5 Migrating Custom Tables (z-Table)	96
5.1 Motivation	96
5.2 Migrating Custom Tables Using an Existing Migration Object.....	96
5.3 Migrating Custom Tables Using an Own Migration Object.....	99
6 Migration Strategies (SAP for Utilities).....	100
6.1 Motivation	100
6.2 Installation Structure History.....	100
6.2.1 Standard Scenario (No Installation Structure History)	100
6.2.2 Extended Device History	101
6.2.3 Extended Billing History.....	102

6.3 Historical Billing Document	103
6.3.1 Migration Objects Relevant for Migrating Billing Documents	103
6.3.2 Easy Bill Correction Framework (EBF)	104
6.4 Historical Rate Data	107
6.5 Historical Consumption	107
6.6 Device Group	108
6.7 Disconnection and Reconnection	108
6.8 Sample Lots	111
6.9 Point of Delivery	111
6.10 Service Provider	112
6.11 Energy Data Management (EDM)	112
6.12 Installment Plan	113
6.13 Dunning History	113
6.14 Collection Agency	114
6.15 Cash Security Deposit	117
6.16 Budget-Billing Plan	117
6.17 Payment Scheme	118
6.18 Replication to SAP CRM	120
6.18.1 Data Model	120
6.18.2 Replication Strategies	120
7 FAQ (Frequently Asked Questions)	122
7.1 Motivation	122
7.2 FAQ Usage of the IS Migration Workbench	122
7.2.1 One or more fields cannot be found in a migration object	122
7.2.2 Why is it required to transfer default information?	122
7.2.3 Error EM 024: Object in import file header does not correspond	122
7.2.4 Error EM 044: Change not possible due to blocking status	123
7.2.5 Warning EM 188: Mandatory indicator deleted	123
7.2.6 Error EM 104: Table x not supplemented	123
7.2.7 How can TEMKSV database table become part of a client copy?	124
7.2.8 Can the rows in TEMKSV database be deleted after go-live?	124
7.2.9 Why to re-generate the load report after a transport request?	125
7.2.10 Data Migration works in one system but not in a second system	125
7.2.11 Why does delta replication not work during data migration?	126
7.2.12 Why does the CRM_REPL_OFF control parameter not work?	127
7.3 FAQ Migration of Addresses	128
7.3.1 Can the address validation be deactivated during data migration?	128
7.3.2 How to deal with uncleansed addresses in test migration cycles	129
7.4 FAQ Migration of Business Partner	130
7.4.1 How to migrate addresses of a business partner?	130
7.4.2 How to change a migrated address of a business partner	131
7.4.3 How to create SD customers while migrating business partner	131
7.4.4 How transfer data for only fields which are subject to be changed?	132
7.4.5 Error R1 140: Specify at least one valid business partner role	132
7.4.6 How to migrate identification numbers of business partners?	133
7.4.7 Migration of direct input structures that are not part of the standard?	133
7.4.8 Error AM 890: Internal Error - Value range of LV_DEF_CNR	139

7.5 FAQ Migration of Contract Account.....	140
7.5.1 Error >3 047: Specify bank details for incoming payment methods.....	140
7.5.2 Error >3 399: Specify a standard company code of contract accounts.....	140
7.5.3 Error >0 425: Field DFKKLOCKS-LOTYP is not ready for input.....	140
7.5.4 Error >1 004: Administrative data not defined for application T.....	140
7.5.5 How transfer data for only fields which are subject to be changed?	141
7.6 FAQ Migration of Financial Documents	142
7.6.1 Error >0 005: No free number range found for document type.....	142
7.6.2 Error >0 200: Posting period for company code already closed	142
7.6.3 Is it possible to create installment plans with external numbering?	142
7.6.4 How to migrate rounding balances?.....	143
7.7 FAQ Migration of Open Item	144
7.7.1 Error >0 009: Reconciliation key already in use	144
7.7.2 Is it possible to reuse reconciliation keys?	144
7.7.3 Migrating open items with division but without contract information?	145
7.7.4 Migrating open items with VAT tax amounts	145
7.7.5 Can migrated financial documents be reversed?	146
7.7.6 Are there any migration specific FI-CA events?	146
7.8 FAQ Migration of Payment.....	147
7.8.1 Why do migrated payments not clear open items?	147
7.8.2 Error due to competing processes	147
7.8.3 Error >0 009: Reconciliation key already in use	148
7.8.4 Is it possible reusing reconciliation keys	148
7.8.5 Is it possible to use the DOCUMENT migration object for payments?.....	148
7.8.6 Error >4 036: "Document does not exist" when paying security deposits	149
7.9 FAQ Meter Reading	151
7.9.1 Creating move-in meter reading during device installation.....	151
7.9.2 Creating different meter reading types during device installation	151
7.9.3 Billing Error "Move-in meter reading missing or incorrect".....	151
7.9.4 Creating meter reading order?.....	152
7.9.5 Migration of meter reading required during device installation/removal?.....	152
7.9.6 Deferred migration of meter reading for a register relationship?	152
7.9.7 Deferred migration of meter reading for rate data changes of devices?	153
7.10 FAQ Creation of Inactive Contracts	154
7.10.1 How suppress billing triggers when migrating inactive contracts.....	154
7.10.2 Error AH385: Contract must be billed first?.....	154
7.10.3 Creating meter reading orders with reason move-out?	155
7.10.4 Creating active contracts with the MOVE_IN_H migration object?	155
7.11 FAQ SAP for Utilities	156
7.11.1 No update of business partner addresses during move-in	156
7.11.2 Migration of multiple time slices for an installation	156
7.11.3 The start date of the device group is set to the system date?	156
7.11.4 PRUEFKLE field remains empty after device installation	157
7.11.5 No migration of custom fields for device locations	157
7.11.6 Error AH 348: The facts are not consistent	157
7.11.7 Error M3 897: Plant data of material is locked.....	158
7.11.8 How to migrate equipments?	158
7.11.9 How to migrate historical service notifications?.....	158

7.11.10 Error IO 151: Error while processing the MASE record?	159
7.11.11 How to use mass processing for BBP_MULT migration object?.....	160
7.11.12 How to use mass processing for SECURITY migration object?	161
7.11.13 How to migrate long texts of functional locations?	161
7.11.14 How to migrate classification data of function locations?.....	162
7.11.15 Advanced Metering Infrastructure (AMI) specific data?	163
7.11.16 How to migrate formula profiles?	164
7.12 FAQ Miscellaneous	167
7.12.1 What are the correct dates for migrating data?	167
7.12.2 Runtime error with condition OBJECT_NOT_FOUND?	167
7.12.3 How to send e-mails and SMS during data migration?.....	168
7.12.4 How to debug a load program.....	168
7.12.5 How to create a migration object to execute an ABAP program?	169
7.12.6 How to Execute Programs in a Group Import?.....	169
8 Migration from SAP System to SAP System	170
9 Appendix with ABAP Code Snippets	171
9.1 Service Function Module for EDM RTP Interface.....	171
9.2 Direct Input Modules for Contract Account (BDT)	172
9.3 Service function Module to Migrate a Custom Table	174
9.4 Function Module to Determine Reconciliation Keys for a Reuse.....	177
9.5 Function Module to Send a Status Email During Data Import.....	178
9.6 Service Function Modules for Service Notifications.....	180
9.7 Service Function Module to Execute a Program	183
10 Getting Started.....	184
10.1 Motivation	184
10.2 Start of the IS Migration Workbench	184
10.3 Creation of a Migration Company.....	185
10.4 Creation of a Migration User.....	187
10.5 Copy of a Migration Object	189
10.6 Maintenance of the Structure Customizing	190
10.7 Maintenance of the Field Customizing.....	193
10.8 Creation of an Import File	195
10.9 Execution of a Data Import	200

1 IS Migration Workbench

1.1 Introduction to the IS Migration Workbench

A migration workbench in an SAP system usually represents the third phase in the extraction, transformation and loading process (ETL process). It is a tool to support the loading or importing of data into the SAP system. SAP has many migration workbenches like batch input, the Legacy System Migration Workbench (LSMW), or the Data Transfer Workbench (DX-WB). The IS Migration Workbench completes these known migration tools. The IS Migration Workbench supports the initial data load of business master data and transaction data into an SAP system; it also supports the synchronization of data during a phased data migration process.

The IS Migration Workbench represents an open and documented interface based on business objects. A data migration using the IS Migration Workbench ensures high quality and consistency of the migrated data. Direct input modules create or change all data in the database, instead of updating the database directly. Almost the same code is executed during data migration as during the creation or change of information using dialog transactions. This includes the available check and verification routines of the respective dialog transactions.

The IS Migration Workbench is available for data migration into the following SAP industry solutions:

- SAP Contract Accounts Receivable and Payable
- SAP for Utilities
- SAP for Telecommunication
- SAP for Insurance
- SAP for Public Sector

For data migration into these SAP industry solutions, the IS Migration Workbench is the primary load tool used.

1.2 Proven Concepts

In the IS Migration Workbench many concepts and techniques are implemented based on experience in migration projects from earlier SAP ERP releases like R/2 and R/3. These are:

- Migration of business objects instead of direct updates in the database tables
- Definition of flexible field rules that can be customized easily
- Generation of load programs including the defined field rules
- Display and print of the interface structure to support the creation of import files
- Execution and monitoring of the data import of high data volumes
- Administration of import files
- The following additional functions:
 - Administration of all keys and relationships between objects in the legacy system and SAP system
 - Usage and transfer of required fields instead of transferring all fields of an interface structure
 - Comprehensive and detailed migration specific documentation of migration objects, interface structures and fields
 - Generation of test data with an import file editor specifically for a support in the test phase
 - Definition of own migration objects

1.3 General Description

The IS Migration Workbench is an open interface for migrating (or loading, or migrating) master data and transaction data into an SAP system. The application is based on the data model in SAP. This means that the data model or the function of the legacy system for data migration is not used. The migration procedure cannot run based on database tables because of the SAP system's relational data model. Instead, it is based on logical units called **migration objects**.

The application transfers the data into the SAP system in these units. The application updates all database tables related to a business object at the same time as creating a business object (for example, a business partner), instead of updating database table by database table. There are special function modules in SAP for the business objects of the application. The ABAP code of these service function modules corresponds to the code of the respective dialog transactions for the same function. This guarantees maximum consistency of the data during its creation. This also makes an object-oriented data transfer possible.

The main technique used during migration is **direct input**. During direct input, no virtual user interaction or terminal I/O (input/output) takes place as during the known batch-input technology (also known as Batch Data Communication). To initiate the direct input migration technique you call a direct input **service function modules** and pass the data found in the import file. There is a specific service function module for each of the migration objects. Using the service function modules leads to a better performance of the load process.

For the data migration, a **load report** (or import program) is required. The load report reads the import file and creates the business objects in the SAP system by passing the data to the direct input function module. SAP does not deliver load reports. The IS Migration Workbench generates the load report directly in the SAP system after finishing or changing the configuration of a migration object.

It is possible to execute the data load in parallel to ensure optimal use of the available system resources. A parallel data load is possible for data imports of the same migration object and for data imports of various migration objects. In the latter case, you must take logical dependencies and locks into consideration. The IS Migration Workbench supports the execution and monitoring of a parallel load with the **distributed import** function. Migration statistics are created for each data load started with the IS Migration Workbench. The migration statistics allows the monitoring of the progress of a data load in displaying information, such as, the number of migrated data objects, number of errors, and throughput per hour.

The application generates an error message if the system cannot create a business object. The application collects all error messages in an **error log** (application log) for an evaluation after the data load has finished. It is possible to evaluate a huge error log or the logs of multiple data loads by creating statistics that show the frequency of all error messages that occurred during the selected data loads.

The IS Migration Workbench administrates the keys and relationships between the business objects in the legacy system and the SAP system. The **key and status management** (KSM) function maintains these relationships. The KSM maintains the status information regarding whether a business object could be migrated successfully or not.

Figure 1-1 shows the main screen of the IS Migration Workbench. The area on top of the screen shows the name of the selected ACCOUNT migration object and the name of the selected VKP structure. It also shows a link to the object and structure documentation. The left column shows the list of the standard migration objects in an SAP system configured for SAP for Utilities. The column in the middle shows the list of available structures of the selected objects. The right column shows the list of customized fields of the selected structure. All functions of the IS Migration Workbench can be reached from here.

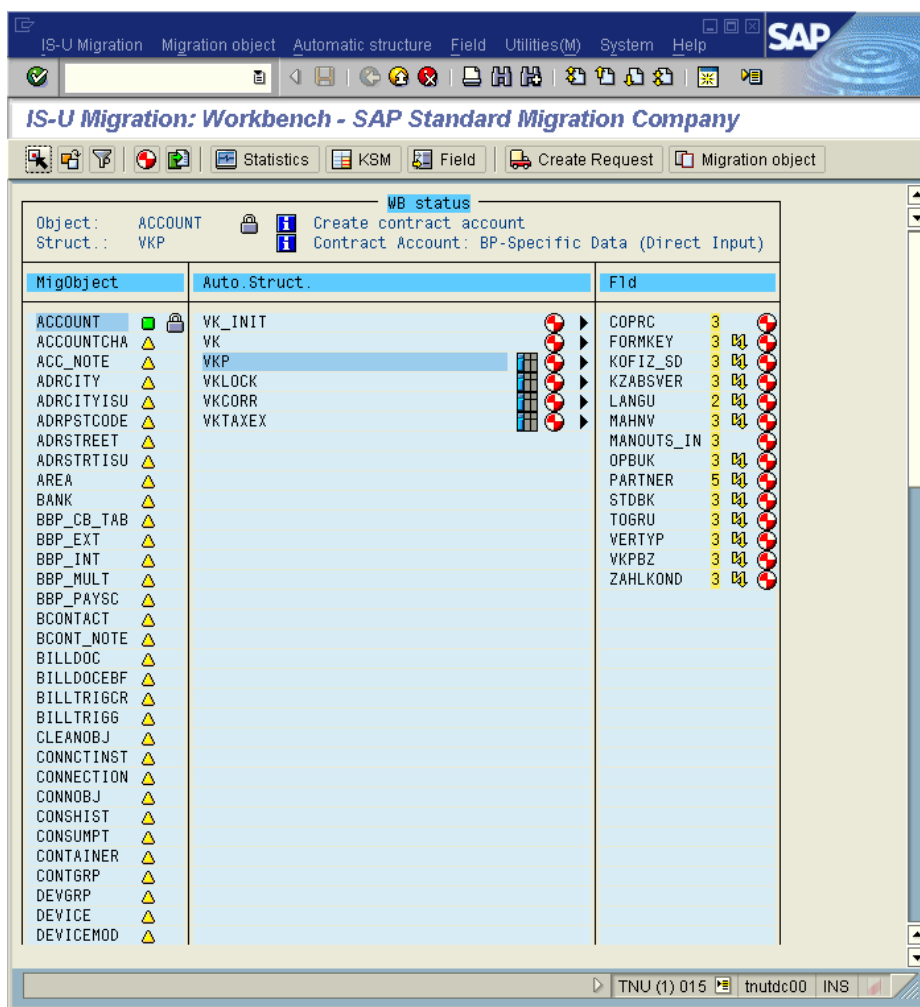


Figure 1-1: Main Screen of the IS Migration Workbench

The displayed symbols on the main screen have the following meaning:

Symbol	Column	Meaning
	MigObject	Load report generated and ready to be used
		Load report needs to be (re-)generated
		Load report cannot be generated due to configuration
		Migration object is locked and can be changed
		Migration specific documentation available (migration object)
	Auto Structure	Structure is used in migration process (<i>generated</i> structure)
		Structure is re-occurring (internal table in automation structure)
		Migration specific documentation available (structure)
1	Field	Field rule <i>Initial Value</i>
2		Field rule <i>Fixed Value</i>
3		Field rule <i>Transfer</i>
4		Field rule <i>ABAP rule</i>




5	Field rule <i>via KSM</i>
6	Field rule <i>Customer Field</i>
9	Field rule <i>Conversion Table</i>
	Field is a mandatory field
	Field is a non-migration field
	Field is used in migration process (<i>generated</i> field)

Figure 1-2: Meaning of the Symbols Displayed in the Main Screen

2 Functions in Detail

2.1 Load Program

SAP does not deliver load programs. The load report for a migration object is generated from the settings in the IS Migration Workbench. The application translates the field rules into ABAP code. You must regenerate the load report after every change to the migration Customizing for a migration object. As soon as the load report is outdated (that is, after settings in the migration object were changed), the previously generated load report becomes obsolete. This is because a regeneration of the load report became necessary to reflect the changes in the load report. A data import is not possible with this status. The header of the report contains the generation time and the comment lines contain the generating user.

There is an option to include your own project-specific ABAP code (**events**) in defined places (**generation points**) of the load report (for more information, see chapter 4 *Enhancing a Load Program*).

2.2 Migration Company

The migration Customizing is a client independent configuration. The client in the SAP ERP system is a legally and organizationally independent unit. This keeps all business data protected from other clients. A three-digit client number identifies clients during login.

The **migration company** represents the project to which you allocate **migration objects**. A migration object is a logical business unit to migrate business objects in the SAP system during data migration (for more information about migration objects, see chapter 3). To allow projects to work with multiple migration configurations, you can create different migration companies and copy migration objects to each of the migration companies. The migration company SAP serves as a reference configuration. SAP delivers the individual migration objects with a basic configuration that can be overwritten when an SAP system upgrade takes place.

To create a migration company or to change the settings, choose *IS Migration → Company Maintenance*. It is mandatory to enter a package (**development class** field) when creating an own migration company. In a package, you group related objects in the ABAP Workbench together. Enter the assignment of an object to a development class in the object directory (TADIR). The package determines the transport layer that defines the transport attributes of an object. We recommend that you define a migration project specific development class in the customer namespace. Using a standard package (for example, the package of the IS Migration Workbench EEMI) is not permitted. An own package is required because the IS Migration Workbench automatically generates ABAP dictionary (DD) objects, function groups, and further ABAP modules that require a package during the creation process.

Figure 2-1: General Parameter of a Migration Company

In the defined migration company, you can define default values for the location of the import file. You can also define the character set you use to create the import files. The **Generic Suffix Length** parameter is important. During an automatic file split (for example, while executing a Data Import), the application generates file names. The names of the files contain the placeholder character &. The application replaces the placeholder & with a consecutive number. You can define the number of digits of the consecutive number using the **Generic Suffix Length** parameter. For more information, see chapter 2.7 *Distributed Import*.

Company		CUST		Last Changed On		KELLERFR 24.09.2008 19:37:46		Status		Saved	
General Data Import											
General											
File name suffix	MIG			Gen. Suffix Length	3						
Leg.sys.char.s.	1100			SAP internal, like ISO 8859-1							
IS-U char. set	4103			UTF-16LE Unicode / ISO/IEC 106							
Dialog Processing											
DIA rem. server	us4479_P7I_79			Windows NT							
DIA mig. path	D:\migration\										
Background Processing											
BTC rem. server	us4479_P7I_79			Windows NT							
BTC mig. path	D:\migration\										

Figure 2-2: Default Parameter for Data Import

2.3 User Parameters

You must define user specific workbench parameters for each **migration user** before using the IS Migration Workbench. For a definition of these user parameters, choose *IS Migration* → *User Parameters*. Figure 2-3 shows the screen to enter the general user parameters, for example, the default migration company.

User		KELLERFR		Last changed by		KELLERFR 19.01.2009 14:18:25		Status		Saved	
General Data import Data exchange Monitoring											
Company											
Company	CUST			<input checked="" type="checkbox"/> Fxd start co.							
<input type="checkbox"/> Other disconn. sts permttd											
Display											
<input type="checkbox"/> No Longer Display											
<input checked="" type="checkbox"/> Display WB status											
<input checked="" type="checkbox"/> Display entire field											
<input type="checkbox"/> Barrier-Free Display											
Number displayed DRs 1.000											

Figure 2-3: General Parameter of a Migration User

The default parameters for the data import are the same as for the migration company. If no respective user parameter is defined, IS Migration Workbench uses the parameter set that is defined in the selected migration company. The user specific parameters take precedence over the migration company parameters.

2.4 Field Rules

The **automation data** of a migration object represents all fields used by a service function module to create a business object in the SAP system. The automation data structure (automation data) consists of one or more structures. In the IS Migration Workbench, a **field rule** describes the unique rule with which an individual field in the automation data structure is processed during data import. You must define a field rule for each field of an automation data structure. The default field rule is the processing of the initial value of the field.

There are fields of an automation data structure that you can switch off. This means that they are not to be populated and are not required in the data migration process. You cannot switch off mandatory fields. The remaining fields form the **customer structure**; this is the customized automation data structure reduced by a number of fields. Every field of a customer structure has to have a rule other than the *Initial* or *Fixed value* field rule. This is because only fields for which you have transferred values in the import file are part of the customer structure. The load program reads the data in the layout of the customer structure, whereas, the record data type (that is, a six characters field in each data record in the import file) can uniquely identify every customer structure.

When you use a set of customized automation data structures, this generates an import process for the necessary fields only. Figure 2-4 shows the upper part of the Field Maintenance screen. An own sub screen exists for each field rule; for some field rules additional parameters may be required. The **Generate** field indicates whether a field is to be used during the migration process. If the field **Required Field** is marked, the load report checks whether the respective field in the automation data has been populated with a non-initial value while processing the field rules. The field **No mig. Field** signals that either the application is ignoring the field, or a value may generate undesired results in the database. In this case, you must not use it in the data migration process.

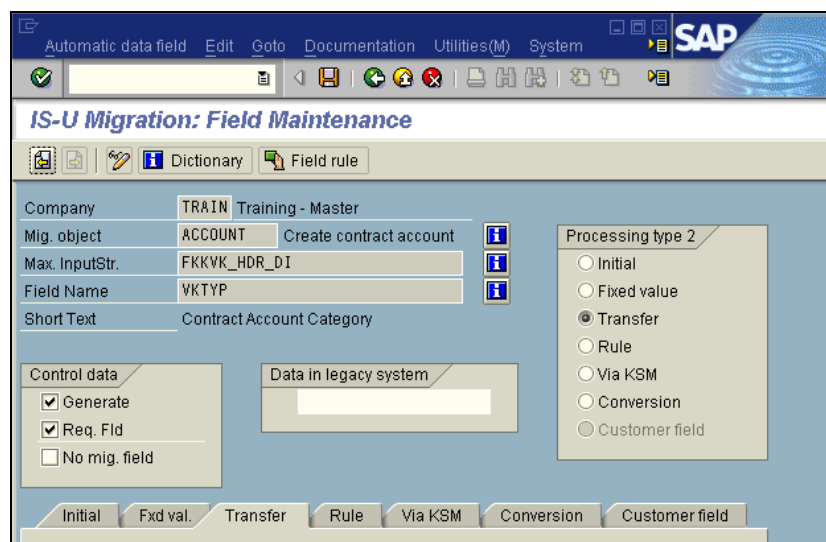


Figure 2-4 Field Maintenance Screen

You can apply the following field rules. They are explained in the subsequent chapters:

- **Initial Value:** Assigns the initial value to the field from the dictionary
- **Fixed Value:** Assigns a constant value to the field
- **Transfer:** Assigns the value as transferred from the legacy system to the field
- **ABAP Rule:** Executes a series of ABAP statements (complex field rules)
- **Via KSM:** Replaces the key of an already imported object with the newkey
- **Conversion:** Maps through a conversion table
- **Customer Field:** Projects specific field

2.4.1 Field Rule Initial Value

For all fields with the *Initial value* field rule, the application assigns the initial value as defined in the ABAP dictionary (DD). The sub screen does not contain further parameters. This is the most common field rule.

A field with the *Initial value* field rule is not part of the customer structure.

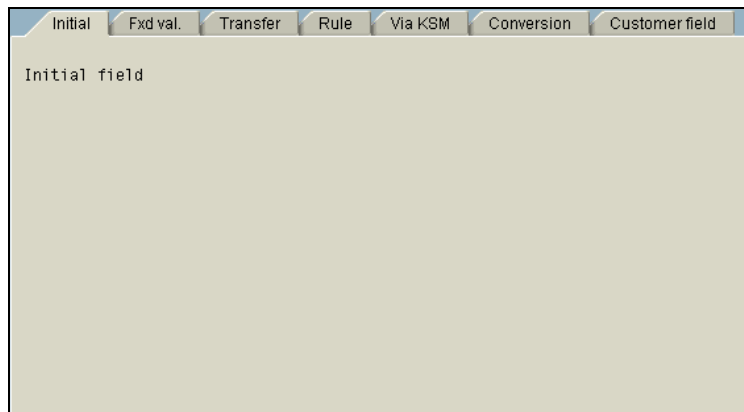


Figure 2-5 Field Rule *Initial Value*

2.4.2 Field Rule Fixed Value

Use the *Fixed value* field rule to assign a constant value to the respective field in the automation structure, as defined by a **fixed value object**. A fixed value object defines the value you assign, which you can re-use for various fields.

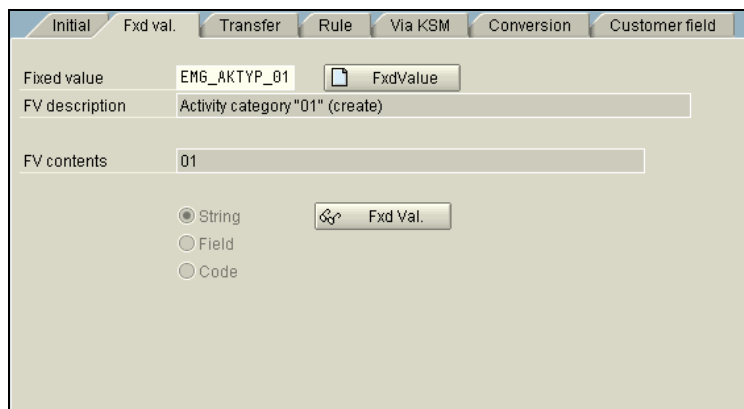


Figure 2-6 Field Rule *Fixed Value*

For a field with *Fixed value* processing type, there is no corresponding field in the customer structure to be processed. Fields with the *Fixed value* field rule do not become part of the customer structure. The correct value for the field in the automation data structure is determined during the processing of the field rule, independent of any data from the legacy system. A fixed value object contains all the necessary parameters to determine the correct value for a field with the *Fixed value* field rule allocated. Figure 2-7 shows the maintenance screen for fixed value objects. You can define a new fixed value object in the sub screen of the field rule (*Create FxdValue*). Alternatively, choose *Utilities* → *Maintain Fixed Value*.

The fixed value object determines the correct value for the field in the automation data structure in the following ways:

- String as a constant value
- System field (like SY-DATUM to determine the system date)
- ABAP code

A string as a constant value is adequate, if one field is always populated with exactly the same value. You can use a system field if the content of a field depends on a user name, or the date or time, the data migration process is executed. An ABAP code is adequate if a field needs to be populated by information that the system must determine by an algorithm. The implemented ABAP code must contain at least a transfer to the symbolic variable **\$\$\$**. For the *ABAP* field rule the symbolic variable **\$\$\$** is used as the placeholder for the field in the automation data structure. The load program generator replaces the symbol with the correct field reference to the field in the automation data structure when it generates the load report. The symbolic variable **\$C\$** cannot be used because a field with a *Fixed value* field rule is not part of customer structure and no information can be transferred to the fixed value object.

The screenshot displays the 'IS-U Migration: Maintain Fixed Value' screen in SAP. At the top, there's a menu bar with 'Fixed value', 'Edit', 'Goto', 'System', and 'Help'. Below the menu, there are icons for various actions like save, delete, and print. The main area is divided into several sections:

- Fixed value:** EM6_FIKEY
- Status:** Modified
- FV description:** Reconciliation key for function module ISU_M_FIKEY_DETERMINE
- For domain:** FIKEY_KK
- Created on:** 05.07.2000
- Changed on:** 11.08.2004
- Created by:** SAP
- Changed by:** SAP

Below these fields is the 'Fixed value definition' section. It has three radio buttons: 'String', 'Field', and 'Code'. The 'Code' radio button is selected. To the right of the radio buttons is a text area labeled 'FV contents' which contains the following ABAP code:

```

ABAP Code
" call function to determine new recon key for user
call function 'ISU_M_FIKEY_DETERMINE'
  importing
    y_fikey      = $$$
  exceptions
    no_temfikey = 1
  
```

At the bottom of the screen, there's a status bar showing 'EMIGFVA', 'us4479', and 'INS'.

Figure 2-7 Fixed Value Object Maintenance Screen

2.4.3 Field Rule *Transfer*

The *Transfer* field rule assigns the value, as transferred from the legacy system in the import file, to the respective field in the automation structure. On the sub screen, the screen displays an input help for the field if one is defined for this field in the ABAP dictionary (for example, search help, check table). The input help enables you to identify possible values that you assign. If the ABAP dictionary defines the field as either a packed or a currency field, you can select whether you use the internal or external representation of a field in the customer structure. We recommend that you always use the external (character) representation because this makes the creation of the correct file information easier.

A field with the *Transfer* field rule is part of the customer structure.

Figure 2-8 Field Rule *Transfer*

2.4.4 Field Rule *ABAP Rule*

Use the *ABAP Rule* field rule to assign a value to the respective field in the automation structure. This field is determined by a value transferred in the import file and is subsequently processed by a custom specific ABAP code. For more information about this field rule, see chapter 4.3.1 *Code on Field Level*.

A *Transfer* field rule is part of the customer structure.

Figure 2-9 Field Rule *ABAP Rule*

2.4.5 Field Rule *via KSM*

Use the *Via KSM* field rule to assign a value to the respective field in the automation structure by replacing the transferred oldkey with the correct SAP internal key. For more information about this field rule, see chapter 2.5 *Key and Status Management*.

A field with the *Via KSM* field rule is part of the customer structure.

Figure 2-10 Field *Via KSM*

2.4.6 Field Rule *Conversion*

The *Conversion* field rule replaces the transferred value in the import file with another value defined by a **conversion object**. A conversion object defines a conversion table to convert a discrete incoming value into a predefined value. You can reuse a conversion object for various fields with a domain identical to the domain of the conversion object. In the ABAP dictionary (DD), a domain describes the technical attributes of a field, such as, the data type or the number of positions in a field.

A field with the *Conversion* field rule is part of the customer structure.

Figure 2-11 Field Rule *Conversion*

A conversion object contains all the necessary parameters to determine the correct value for a field when you allocate the *Conversion* field rule. Figure 2-12 shows the maintenance screen for conversion objects. You can define a conversion object in the sub screen of the field rule (*Create Conversion Object*). Alternatively, choose *Utilities* → *Maintain Conversion Object*.

The screenshot shows the 'IS-U Migration: Maintain Conversion Object' window. The 'Conversion values' button is highlighted with a red rectangle. The form contains the following data:

Field	Value
Conversion obj.	R_TITLE
Status	Saved
Conv. obj. text	Title Business Partner
For domain	AD_TITLE
Created on	19.09.2003
Changed on	19.09.2003
Created by	KELLERFR
Changed by	KELLERFR

Input Settings:

Field	Value
Input type	C
Input length	6
Upper/lower c.	<input checked="" type="checkbox"/>

Output Settings:

Field	Value
Output type	C
Output length	4
Upper/lower c.	<input type="checkbox"/>

Error Handling:

- ☒ Display error msg.
- ☐ Set initial value
- ☐ Use default
- ☐ Use input value

Default value:

☐ Do not conv. init. val

Footer: EMIGCNV | tnutdc00 | INS

Figure 2-12 Conversion Object Maintenance Screen

The **Input Type** (character, numeric) field and the **Input Length** field define the attributes of the field in the customer structure. You must mark the **Upper/Lower character** indicator, when the text in the field of the customer structure may contain lower case characters. The application allows the definition of lower case letters as input values for the conversion table of the conversion object only if the indicator is marked. The domain assigned to the conversion object defines the attributes of the output values. Choose the **Conversion values** button to maintain the values of the conversion table, or choose *Edit* → *Conversion Value*. Figure 2-13 shows the maintenance screen for conversion values. One of the following error handling procedures can be defined in case an input value cannot be found in the conversion table:

- Display error message
The load program raises the error message EM 066 *Conversion not possible* and the application rejects the data object.
- Set initial value
Instead of the transferred value, the initial value, as defined for the domain of the conversion object in the ABAP dictionary (DD), is assigned to the field in the automation data structure. This action does not raise an error message and the processing of the field rules continues.
- Use given default value
Instead of the transferred value, the application assigns the value, as defined in the **Default value** field, to the field in the automation data structure. This action does not raise an error message and the processing of the field rules continues.
- Use input value
The application assigns the transferred value to the field in the automation data structure. This action does not raise an error message and the processing of the field rules continues.

If the **Do not convert initial value** indicator is marked and in the import file an initial value, as defined for the domain of the conversion object in the ABAP dictionary (DD), has been transferred, the same initial value is assigned to the field in the automation data structure.

The conversion values can be defined using the search help for the domain (if one is defined) to simplify the definition. For an upload or download, choose *Edit* → *Conversion Values* → *Upload* or *Download*.

Conversion input	Conversion output
Cmp	0003
Co	0003
Comp	0003
MR	0002
MS	0001
Mr	0002
Mr.	0002
Ms	0001
Ms.	0001

Figure 2-13 Conversion Object Maintenance Screen Conversion Values

2.4.7 Field Rule Customer Field

A field with the *Customer Field* field rule is known only in the IS Migration Workbench. For more information, see Chapter 4.4.1 *Additional Fields in the IS Migration Workbench*.

A field with the *Customer Field* field rule is part of the customer structure.

Figure 2-14 Field Rule *Customer Field*

2.5 Key and Status Management

2.5.1 General Description

In data migration, the legacy system identifies the business object as a **legacy system key (oldkey)**. SAP identifies this migrated business object as an **SAP key (newkey)**. A function called **key and status management (KSM)** is implemented in the IS Migration Workbench for the following reasons:

- Most of the business objects require a reference to an existing business object (superior object). When creating this business object, it becomes a dependant object (for example, a contract account requires the creation of an existing business partner). However, the legacy system does not know the business partner key in SAP system after the migration of the superior object.
- When creating business objects in the SAP system, it is not possible to use the same key as in the legacy system. Using the oldkey as a newkey (this is external numbering in SAP) depends on whether the oldkey fulfills the requirement of the corresponding newkey for the specific object (for example, the maximum length of the identification of an SAP business partner is defined with 10 characters).
- The need of a mechanism to prevent duplication of business objects when the business object key is determined in the SAP system (internal numbering).
- Implementation of a mechanism to allow a restart of an interrupted or finished data load using the same import file as a source.

2.5.2 Implementation

The key and status management function works according to the following rules:

- Every data object imported has to have a unique legacy system key (or oldkey).
- For each business object (for example, business partner) there is a legacy key and a newkey after a successful import.
- In a migration database table (**TEMKSV** database table), the relationship between the legacy key and the SAP key is stored.
- The existence of the stored relationship represents status information (object already imported or not yet imported).
- The stored information is used as a referencing link to replace a transferred oldkey that has a corresponding newkey during the data load.
- The primary key of the **TEMKSV** database table consists of the client fields, migration company, migration object, and oldkey.

During the data load, the load program executes the following:

1. Read the next record from the import file.
2. Check whether the oldkey already exists in the **TEMKSV** database table. If it does, write an error message and read the next record from the import file.
3. Find fields in the automation data structure that are customized with a *via KSM* processing type.
4. Exchange the oldkey with the corresponding newkey from the **TEMKSV** database table. If there is no entry, write an error message and read the next record from the file.
5. Create business objects in the SAP system. If the creation fails, write an error message and read the next record from the file.
6. Store the newkey of the new business object with the oldkey in the **TEMKSV** database table.

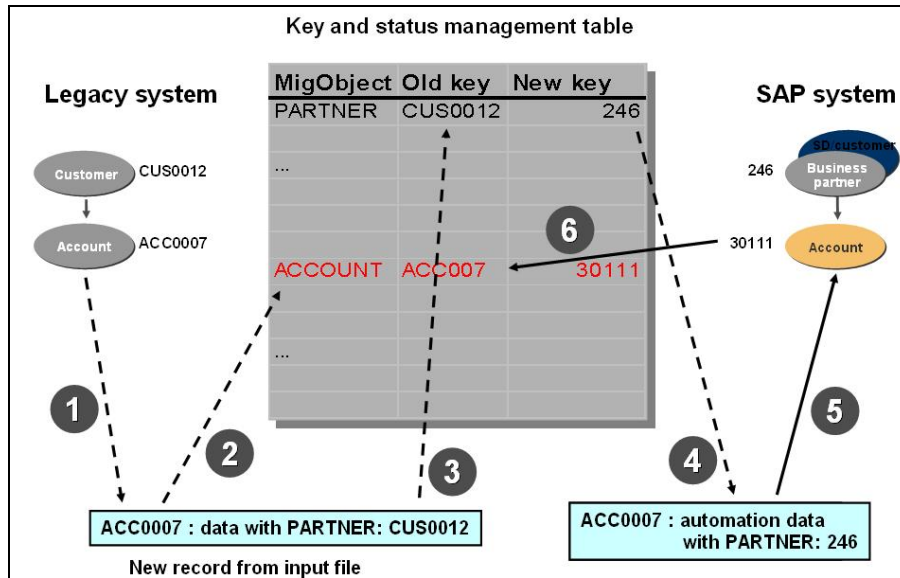


Figure 2-15 Management of KSM During Data Load

2.5.3 Use of the KSM in Field Rules

The *via KSM* field rule is an important field rule. Figure 2-16 shows the maintenance screen for this field rule.

Company: CUST Migration Company Customer

Mig. object: ACCOUNT Create contract account

Max. InputStr.: FKKVKP_S_DI

Field Name: PARTNER

Short Text: Business Partner Number

Processing type 5

☐ Initial

☐ Fixed value

☐ Transfer

☐ Rule

☒ Via KSM

☐ Conversion

☐ Customer field

Control data

☒ Generate

☒ Req. fld

☐ No mig. field

Data in legacy system

Initial Fxd val. Transfer Rule Via KSM Conversion Customer field

MigObject 1: PARTNER Create: Business Partner

MigObject 2

MigObject 3

MigObject 4

MigObject 5

KSA control

☐ Gen. KSM indic.

☐ Gen. Obj. Field

Object Lock

☐ Check lock

☐ Lock Object

Figure 2-16: Field Maintenance Screen with Field Rule Via KSM

As described, the *Via KSM* field rule replaces the oldkey transferred in the import file. The field rule uses the selected migration company, migration object, and the oldkey to lookup the reference link to the Newkey in the TEMKSV database table. It is possible to configure as a cascaded lookup. This means that if the first lookup using the first migration object fails, the application tries lookups with more migration objects until it finds a reference link in the KSM, or an error is raised.



Business partners are migrated with two migration objects, PARTNERRES (residential customer) and PARTNERCUI (commercial & industrial customer). They are based on the same migration object PARTNER but with different migration Customizing. If you enter migration object PARTNERRES (migration object 1) and PARTNERCUI (migration object 2) in the object list, the field rule first checks the combination: migration company / PARTNERRES / oldkey. If it cannot find a newkey, the field rule checks the combination: migration company / PARTNERCUI / oldkey.

2.5.4 Planning for Unique Oldkeys

Use an oldkey that is easy to trace back to the legacy system. This should not pose a problem for most objects. Most legacy systems have a unique number for customers and you can use this as the oldkey. Sometimes the SAP master data object account (where we store financial relevant data) has attributes of the customer object in the old system. This is not an object with a separate key.

In this case, you can use a strategy to have the extract program add an identifier to the legacy customer number. If the legacy customer number for a particular customer is 12345, then the oldkey for the business partner could be BP12345 and the oldkey for the contract account be CA12345. With this numbering, it is simple to trace the oldkey back to the legacy system.

Depending on the data model in the legacy system, you may need schemes that are more sophisticated. Build the plan for the keys early on by naming them in a way that they support later analysis of the loaded data.

2.6 Data Import

2.6.1 Import File

The import file is a sequential file where you keep the data records of a migration object. The import file is a continuous binary stream of concatenated data records. The load program processes the import file one data record at a time. The execution of the field rules populates the automation data structure. Figure 2-17 shows the data flow from the legacy system to the automation data structure.

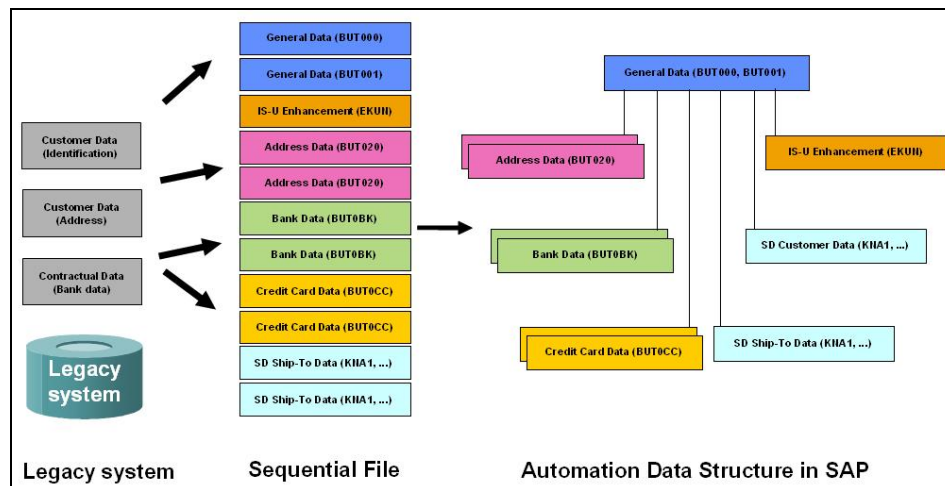


Figure 2-17 Data Flow from the Legacy System to the Automation Data Structure

Each data record has a predefined fixed-length header that transfers the length of each data record. Neither end-of-record signs nor fill bytes at the end of a data record are required for a separation of the data records. The end of the data stream is indicated by either the physical

or the logical end of the import file (depending on the operating system) or by the byte sequence FFFF_x (hexadecimal). All data record in the import file must have the format as shown in Figure 2-18.

Field	Length	Format
Record length	2	Binary
Oldkey	30	Character
Record data type	6	Character
Data	Variable	Character

Figure 2-18: Structure of a Data Record in an Import File

The length of a data record includes the length of the field record (2 bytes). An automatic code page translation (from the code page of the characters in the import file to the code page of the characters in the SAP system) takes place during the data load. As a result, the character portion can be transferred in any code page that is available in the SAP system. The application calculates the length of a data record based on the number of transferred bytes, even though the characters of a certain code page may occupy more than 1 byte per character (for example, UTF-16 Unicode). You must define the codepage of the characters in the import file in the migration company. However, you may overwrite the definition of the codepage in the user parameters or individually for each migration object.

The first data record in an import file is the header record transferring &INFO in the *Record Data Type* field. It can only appear once in an import file. The data section of the header record includes the fields: migration company, migration object, creator of data file, and creation date and time. The **TEMINFO** structure in the ABAP data dictionary defines the structure of the data. Data belonging to one data object is transferred in subsequent data records. You can start a data import only with a migration object whose name and migration company matches to the information in the &INFO data record of the import file. This is to prevent a data import of an import file you have created for a different migration object.

A data record with the &ENDE record data type must close each sequence. After reading this data record, the data is transferred to the application by passing the filled automation data to the defined service function module. To display the correct sequence choose *Utilities* → *Structure Display* → *Display Customer Structure* and then push the button.

Example file for migration object CUST PARTNER	
Lg	Conts.
...	5...10...5...20...5...30...5...40...5...50...5...60...5...70...5...80...5...90...5...100 more
004F	*****&INFOCUST%PARTNER%KELLERFR%YYMMDDhhmmss
0035	*****OLDKEY*****INIT%SSSSSTUUUUGSS
0056	*****OLDKEY*****EKUN%SSSSTTUUSTTTTTTTTTTTTTTTTTTTTTTUSSSTUUUUUUUUSSSS
019C	*****OLDKEY*****BUT000SSSSSSSSSSSSSSSSSSSTTTTTTTTTTTTTTTTTTTTTTUSSSSSSSSSSSSSSSSSS >
004E	*****OLDKEY*****BUT0BKSSSSTTUUUUUUUUUUUUUSSSSSSSSSSSSSSSSSS
004E	*****OLDKEY*****BUT0BKSSSSTTUUUUUUUUUUUUUSSSSSSSSSSSSSSSSSS
0008	*****OLDKEY*****BUT020SSSSSSSSSSSSSSSSSSSTTTTTTTTTTTTTTTTTTTTTTUUUUUUUUUUUUU >
0008	*****OLDKEY*****BUT020SSSSSSSSSSSSSSSSSSSTTTTTTTTTTTTTTTTTTTTTTUUUUUUUUUUUUU >
0053	*****OLDKEY*****SHIPT0SSSSSSSSSSSSSSSSSTTTTTTTTTTTTTTTTTTTTTTUUUUUUUUUUUUU
003C	*****OLDKEY*****TAXNUMSTTTTTTTTTTTTTTTTTTTTTTU
003C	*****OLDKEY*****TAXNUMSTTTTTTTTTTTTTTTTTTTTTTU
0062	*****OLDKEY*****BUT0IDSSSSSSSSSSSSSSSSSSSTTTTTTTTTTTTTTTTTTTTTTUSSSSSSSSSSSSSSSSSS
0062	*****OLDKEY*****BUT0IDSSSSSSSSSSSSSSSSSSSTTTTTTTTTTTTTTTTTTTTTTUSSSSSSSSSSSSSSSSSS
0031	*****OLDKEY*****BUT0ISSTTTTTTTTTTT
0031	*****OLDKEY*****BUT0ISSTTTTTTTTTTT
0026	*****OLDKEY*****&ENDE%
0035	*****OLDKEY*****INIT%SSSSSTUUUUGSS
0056	*****OLDKEY*****EKUN%SSSSTTUUSTTTTTTTTTTTTTTTTTTTTTTUSSSTUUUUUUUUSSSS
019C	*****OLDKEY*****BUT000SSSSSSSSSSSSSSSSSSSTTTTTTTTTTTTTTTTTTTTTTUSSSSSSSSSSSSSSSSSS >
004E	*****OLDKEY*****BUT0BKSSSSTTUUUUUUUUUUUUUSSSSSSSSSSSSSSSSSS
004E	*****OLDKEY*****BUT0BKSSSSTTUUUUUUUUUUUUUSSSSSSSSSSSSSSSSSS
0008	*****OLDKEY*****BUT020SSSSSSSSSSSSSSSSSSSTTTTTTTTTTTTTTTTTTTTTTUUUUUUUUUUUUU >


Figure 2-19: Sequence of Data Records in an Import File

2.6.2 Import File Transformation

Sometimes, in the legacy or transformation system, it is not possible to create an import file in the required binary file format. For a transformation of a text file to the binary file format of an import file, choose *Migration Object* → *Data import* and then *Utilities* → *Convert migration file* or execute the report REMIG_FILE_TRANSFORM. You have the following options:

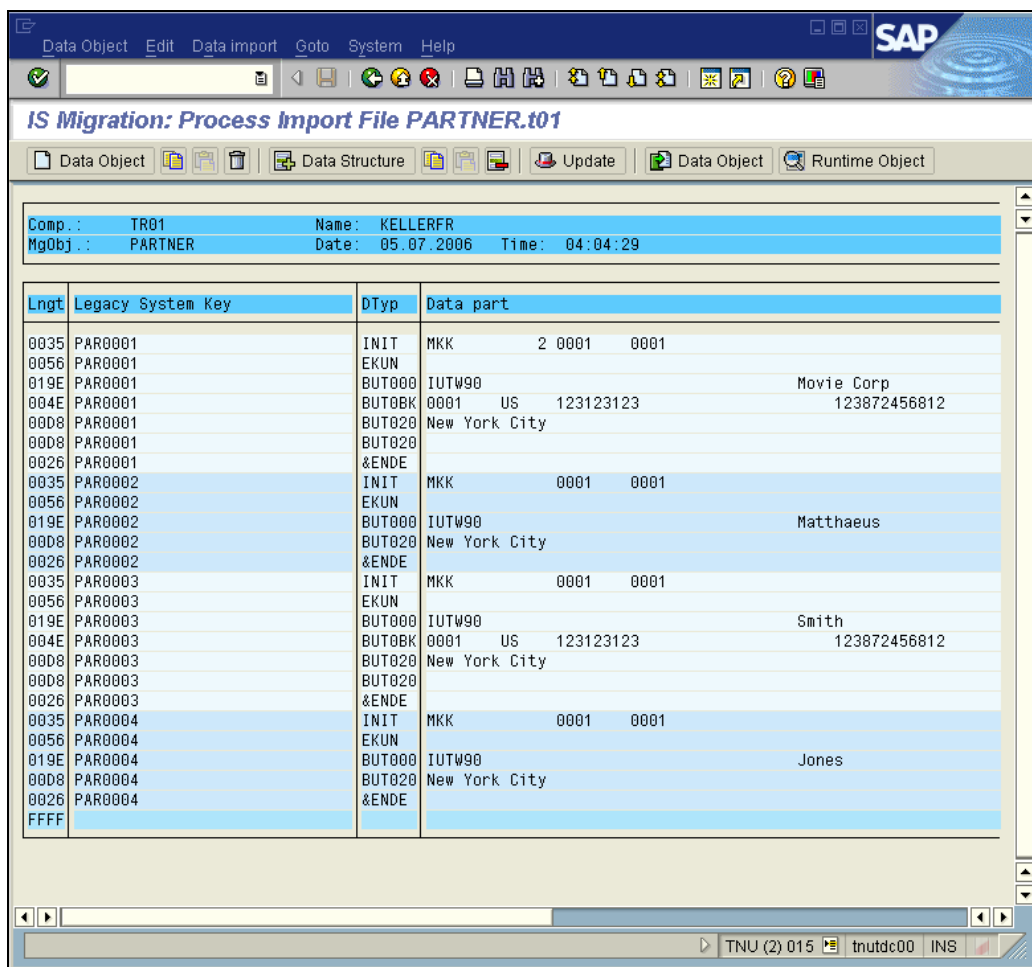
- Transformation of a text file located in the SAP file system
- Transformation of a text file located on your notebook or desktop
- Transformation of a binary file located in the SAP file system. This file is in the correct binary file format except for a 4 byte record length field instead of a two byte record length field

Alternatively, you may upload a text file from your desktop into the import file editor. In the import file editor, choose *Edit* → *Data* → *Upload*.

 You must separate the information in the text file by a horizontal tab stop character between the fields; the text file must not contain a header record. The transformation program writes the header record based on the entered name of the migration object.


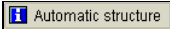

2.6.3 Import File Editor

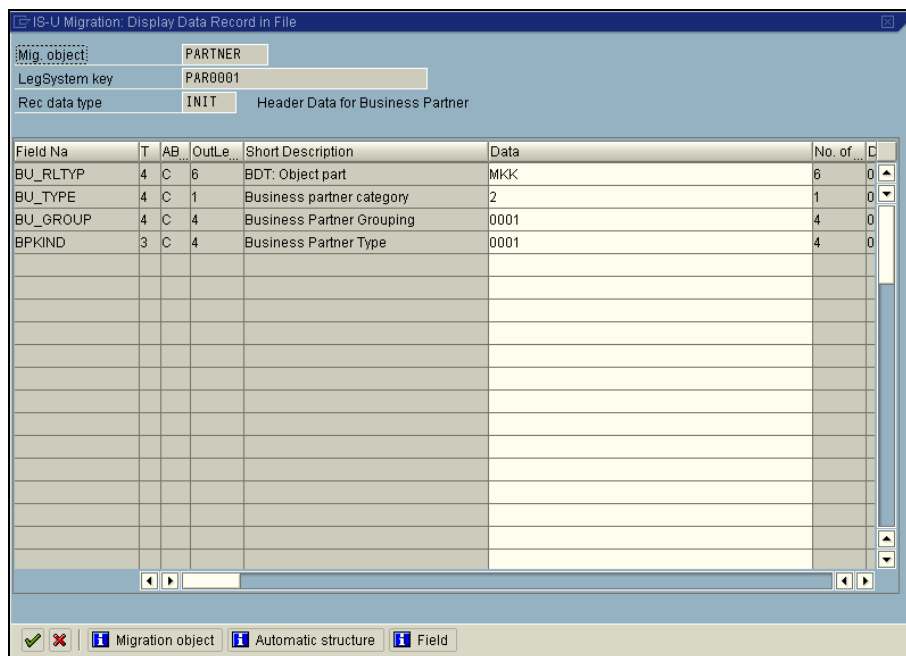
Figure 2-20 shows the available import file editor in the IS Migration Workbench. The editor is available on the data import screen (choose *Migration Object* → *Data import*). The import file editor allows the analysis, creation, and change of data in the import file data based on the Customizing of the migration object.



Lngt	Legacy System Key	DTyp	Data part
0035	PAR0001	INIT	MKK 2 0001 0001
0056	PAR0001	EKUN	
019E	PAR0001	BUT000	IUTW90 Movie Corp
004E	PAR0001	BUT00K	0001 US 123123123 123872456812
0008	PAR0001	BUT020	New York City
0008	PAR0001	BUT020	
0026	PAR0001	&ENDE	
0035	PAR0002	INIT	MKK 0001 0001
0056	PAR0002	EKUN	
019E	PAR0002	BUT000	IUTW90 Matthaeus
0008	PAR0002	BUT020	New York City
0026	PAR0002	&ENDE	
0035	PAR0003	INIT	MKK 0001 0001
0056	PAR0003	EKUN	
019E	PAR0003	BUT000	IUTW90 Smith
004E	PAR0003	BUT00K	0001 US 123123123 123872456812
0008	PAR0003	BUT020	New York City
0008	PAR0003	BUT020	
0026	PAR0003	&ENDE	
0035	PAR0004	INIT	MKK 0001 0001
0056	PAR0004	EKUN	
019E	PAR0004	BUT000	IUTW90 Jones
0008	PAR0004	BUT020	New York City
0026	PAR0004	&ENDE	
FFFF			

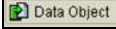
Figure 2-20: Import File Editor

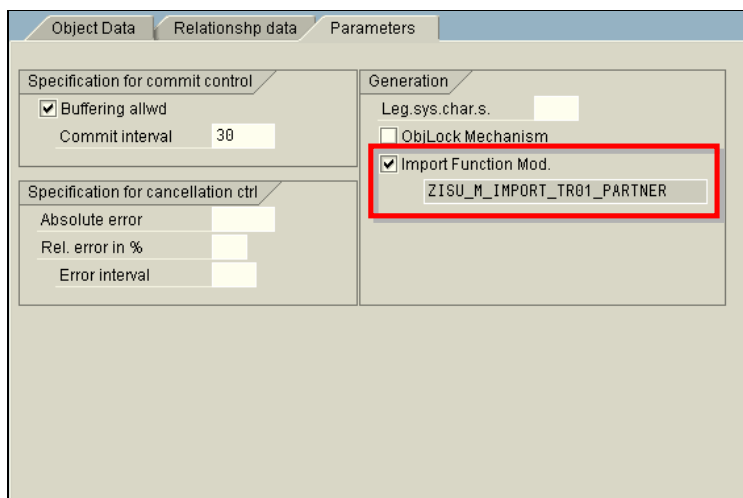
Figure 2-21 shows the screen to enter and change a data record based on the migration Customizing of the `INIT` structure of the `PARTNER` migration object. You can use the `F4` key to display a list of possible entries if a search help is defined in the ABAP data dictionary for the respective field. Push the  button to display the migration object specific documentation and the  button for the automation structure documentation. Push the  button for a migration specific documentation on field level.



Field Na	T	AB	OutLe	Short Description	Data	No. of	
BU_RLTYP	4	C	6	BDT: Object part	MKK	6	0
BU_TYPE	4	C	1	Business partner category	2	1	0
BU_GROUP	4	C	4	Business Partner Grouping	0001	4	0
BPKIND	3	C	4	Business Partner Type	0001	4	0

Figure 2-21: Editing a Data Record

It is also possible to import data objects individually by positioning the cursor on a data object and then pushing the  button. This button is only available if the **Import Function Module** field on the **Parameters** sub screen, which is in the **Object Maintenance** screen, has been marked (see Figure 2-22). The import function module is generated by the IS Migration Workbench when it generates the load report of the migration object. An error message pointing to the cause of the problem is displayed immediately if the data import fails. The main advantage of importing data objects from the import file editor is that you can verify the validity of the data in the import after a change in quick try-and-error rounds.



Object Data Relationship data **Parameters**

Specification for commit control

☒ Buffering allwd
Commit interval: 30

Specification for cancellation ctrl

Absolute error:
Rel. error in %:
Error interval:

Generation

Leg. sys. char. s.:
☐ ObiLock Mechanism
☒ Import Function Mod.
ZISU_M_IMPORT_TR01_PARTNER

Figure 2-22: Parameter to Enable Data Import in the Import File Editor

2.6.4 Data Import

A data import of import files is initiated either from the data import screen (choose *Migration Object* → *Data import*) or with the function mass import (for more information, see chapter 2.7 *Distributed Import*). Figure 2-23 shows a typical data flow diagram of a data migration process with the data import as the central function.

The load program is generated from the settings in the IS Migration Workbench. During a data import, a load program transfers data objects of the import file through the interface structures (automaton data structure) and through the direct input service function modules to the application. The application verifies the data and creates or changes the requested business object and signals the load program that the data was processed successfully. In this case, the load program updates the database tables of the key and status management accordingly. For more information about the key and status management function, see chapter 2.5 *Key and Status Management*.

If the application cannot process the data object due to an error, it signals an error to the load program and returns an error message that describes the encountered problem. The load program saves the error message in an error log and, if requested, writes the erroneous data object in an error file. You can use the error file for a restart of the data load, after the reason for the error has been identified and removed.

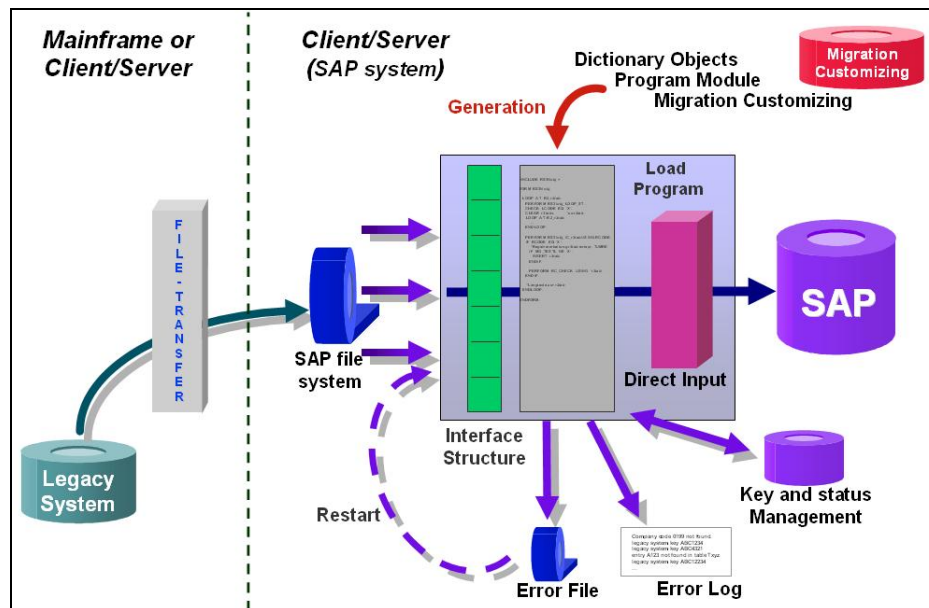


Figure 2-23: Data Flow from the Legacy System to the SAP System

From the data import screen, import files are imported individually. More functions are available from this transaction, such as:

- Creation and deletion of import files
- Split of an import file into smaller files with a defined number of data objects per split file (select *Utilities* → *Break down migration file*)
- Creation of a new import file when searching for data objects in an import file (select *Utilities* → *Search data objects*)
- Conversion of a text file into the binary format of an import file (select *Utilities* → *Convert migration file*)
- Various transactions to monitor the SAP system activity select *Goto*)

2.6.5 Migration Statistics

A migration statistic is created for each data load started with the IS Migration Workbench. The statistics contain the essential information about the import run and allow you to monitor the progress of a data load by displaying information, such as, number of migrated data objects, number of errors, and throughput per hour. Access to the statistics is available from the main screen of the IS Migration Workbench and from various other transactions. You can select the migration statistics to be analyzed with a selection-screen based on many criteria (see Figure 2-24).

IS-U Migration: Analysis of Migration Statistics

Select user and migration object

User Name: CTMIGRA to

Company to

Migration Object to

Select Starting Time

Start Date to

St. Time From: 00:00:00 To: 00:00:00

Select environment and import category

Display All ...

☐ Import runs in batch

☐ Import runs in dialog

☒ Imports in batch and dialog

Only Display ...

☐ Restarts

☐ Imports with Function Module

☐ Single jobs, distrib. import

Other selections

Client to

Migration file name to

Select status of import run

Displaying all import runs with status...

☒ Import finished

☒ Import running

☒ Import Started

☒ Terminated in the Background

☒ Terminated Due to Data Categ.

☒ Terminated due to abs. error

☒ Terminated due to rel. error

☒ Stopped Distributed Import

Figure 2-24: Selection-Screen Migration Statistics

Figure 2-25 shows a list of statistics records during a migration test cycle (migration company is made anonymous with a bar for privacy reasons). The list shows a typical sequence of import runs with various errors.

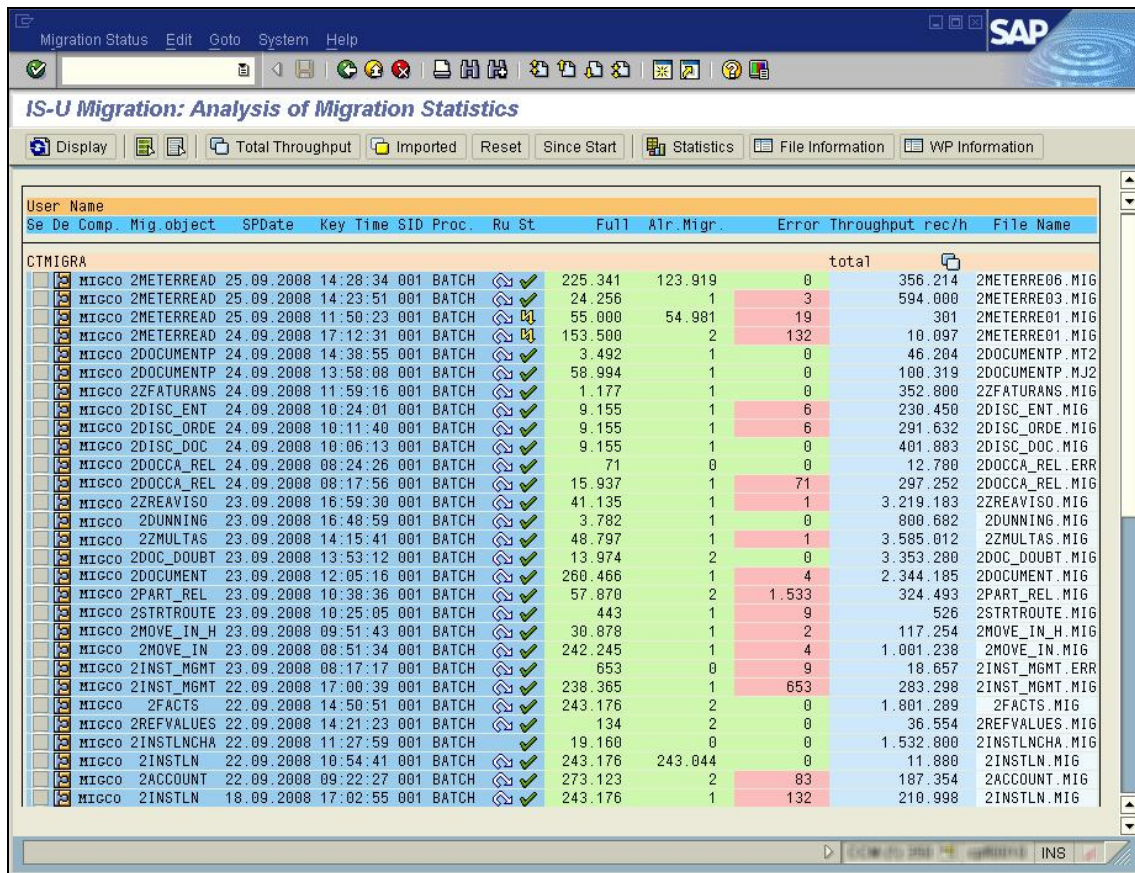


Figure 2-25: Migration Statistics

The displayed symbols in the migration statistics have the following meaning:

Symbol	Meaning
	Import finished
	Import active
	Import started in restart mode
	Job was cancelled in background
	Import cancelled due to a wrong order of data records
	Distributed import stopped by user
	Distributed import finished with cancelled slave jobs
	Attributes of import

Figure 2-26: Meaning of the Symbols Displayed in the Migration Statistics

You can analyze a migration statistic individually by pushing the button. Figure 2-27 shows an error log of an import run with the migration object for a device installation (INST_MGMT migration object). Each error message is completed by the error message EM 100 *Error during processing of legacy system key <oldkey>*, showing the oldkey of the data object for which the error occurred. Click the button to display the longtext of the selected error message.

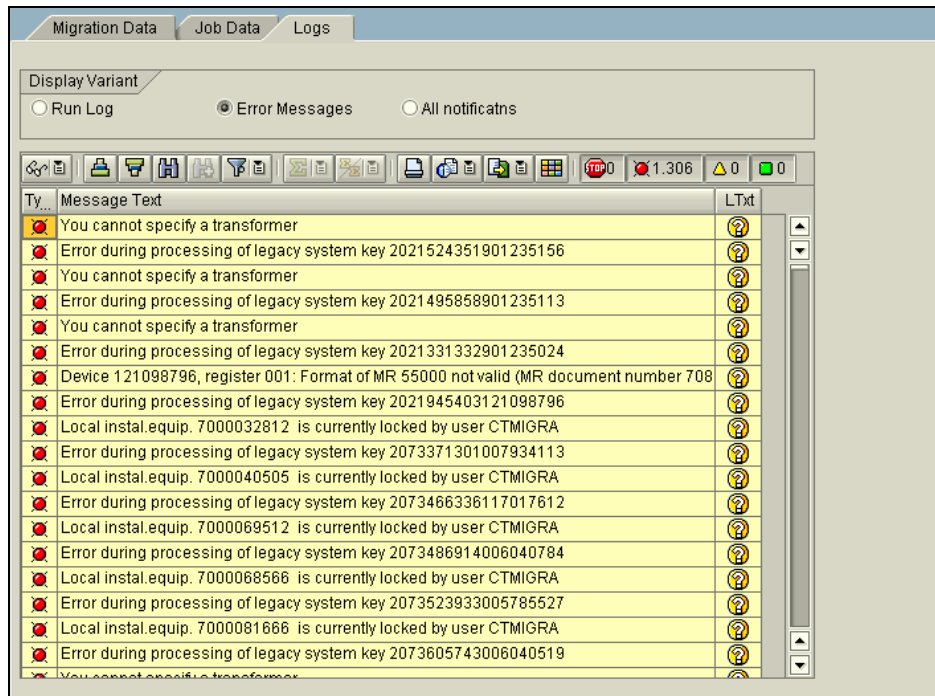



Figure 2-27: Error Log of an Import Run

For the selected error logs, you can create a message statistic to show the error messages and the frequency of their occurrences. To analyze each of the error messages further, create new statistics based on the variable parts of the error message (message variables). To create statistics for more than one import run, select the import runs that you want to create statistics for and push the **Statistics** button. This allows an efficient error analysis for many import runs of the same migration object with many import files. Figure 2-28 shows the statistics of the error log shown in Figure 2-27. To download the created the statistics to your desktop, choose *Statistics* → *Save to PC file*. The created text file separates the statistics information by a horizontal tab stop character in the tab-delimited format. You can upload the file into EXCEL for a further processing.

Error Messages			
Migration Stats			
No.	Message Text	Message ID	Message No
644	&2 &3 &4 is currently locked by user &1	E9	29
7	You cannot specify a transformer	EN	237
2	Device &1, register &2: Format of MR &3 not valid (MR document num	EL	355

Figure 2-28: Statistics of an Error Log

 There is no error log available for an analysis of a canceled import run. If neither the job log nor the short dump (transaction ST22) indicates what might have caused the cancellation, you can transfer all the error messages of the error log to the job log. The **Message in Job Log** user parameter on the **Monitoring** sub screen should be marked for this purpose only due to the negative impact on the performance. To display the job log, select the display variant **Run Log** instead of **Error Messages**. Alternatively, push the **Job Log** button on the **Job Data** sub screen.

The load program writes only the last error message of the application to the error log even the application raised more than one error message while processing the transferred data. Sometimes it is difficult to infer the root of the problem from a single error message. To force the load program to write all created error messages to the error log instead of only the last one, create an event as shown in Figure 2-29. The application writes now all error messages to the error log. For more information about how to create an event, see chapter 4.3.3 *Code on Report Level*.




 The load program cannot differentiate between application internal messages and messages that are relevant for a resolution of the problem. This is why the load program writes only the last error message to the error log. After implementing the event; many error messages can be written to the error log and need to be analysed carefully for their relevance.

Figure 2-29: Event to write all Error Messages to the Error Log

 After a data migration of installation facts for an installation with INSTLN migration object you find in the error log the message AH 348 *The facts are not consistent*. You cannot infer the erroneous fact (or operand) from this general error message. An in-depth analysis shows that the application raised the error AH 423 *Error in operand X (fact group Y, season Z)* which would have allowed to identify the erroneous data in the import file. Because this error message was the 2nd last raised by the application it was not written to the error log.

 Not all error messages in the error log help to find the error in the transferred data or Customizing. After the implementation of the event as shown above the following error messages can be found in the error log: E7 002 *No contract found for Installation X*, E9 014 *Error in reading table ETTIFN for installation X*, AH 423 *Error in operand X (fact group Y, season Z)*, AH 348 *The facts are not consistent* and finally the obligatory message EM 100 *Error during processing of legacy system key X*. The first two error messages can be ignored because they are application internal error messages while the 3rd error message helps to identify the erroneous data causing the problem.

2.7 Distributed Import

2.7.1 General Description

The usage of all available system resources during a data import is important to maximize the data migration performance (number of created objects per hour) and to reduce the necessary data migration window. One option is to execute the data import with many jobs migrating objects in parallel. For a parallelized data import, the import file must be split into smaller import files and one job per split import file has to be scheduled on the available application servers of the SAP system. The **Distributed Import** function supports the configuration and execution of a parallelized data import. Always select the distributed import function unless the data volume is very small. To access the function, choose *Utilities* → *Schedule Import Jobs* → *Distributed Import*.

2.7.2 Implementation

Figure 2-30 shows how the distributed import works. The process is as follows:

1. Start a distributed import to schedule a master job.
2. The master job splits the required files for the first wave of load programs. The number of business objects per file is defined. After the split, the master job schedules the load reports (slave jobs) according to the defined job distribution.
3. The master job continues the file split until the entire import file is split.
4. The master job starts monitoring the scheduled slave jobs (every 5 seconds).
5. The master job schedules a further slave job after a previously scheduled slave job finishes. During this time, the user can change the job distribution online.
6. When the slave jobs have finished, the master job collects the error files into one error file, and deletes the split files and the error files of the slave jobs. The error logs of the slave jobs remain unchanged.
7. The master job finishes.

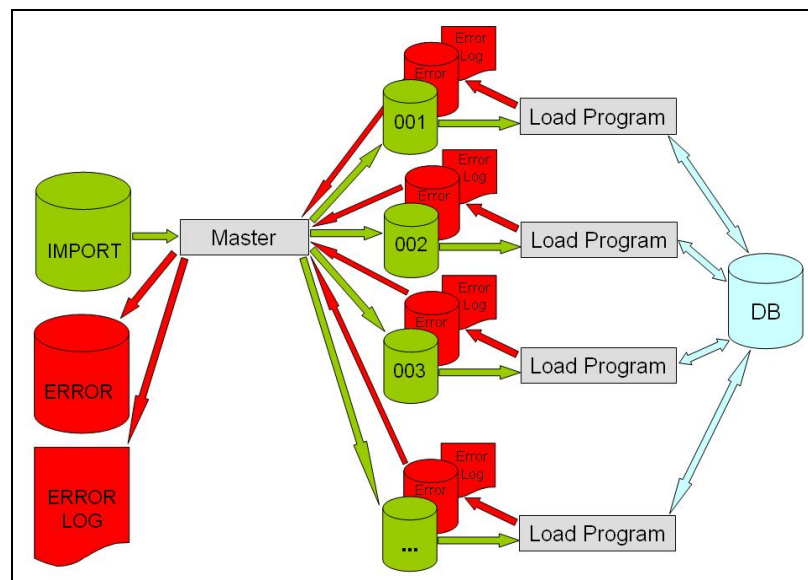
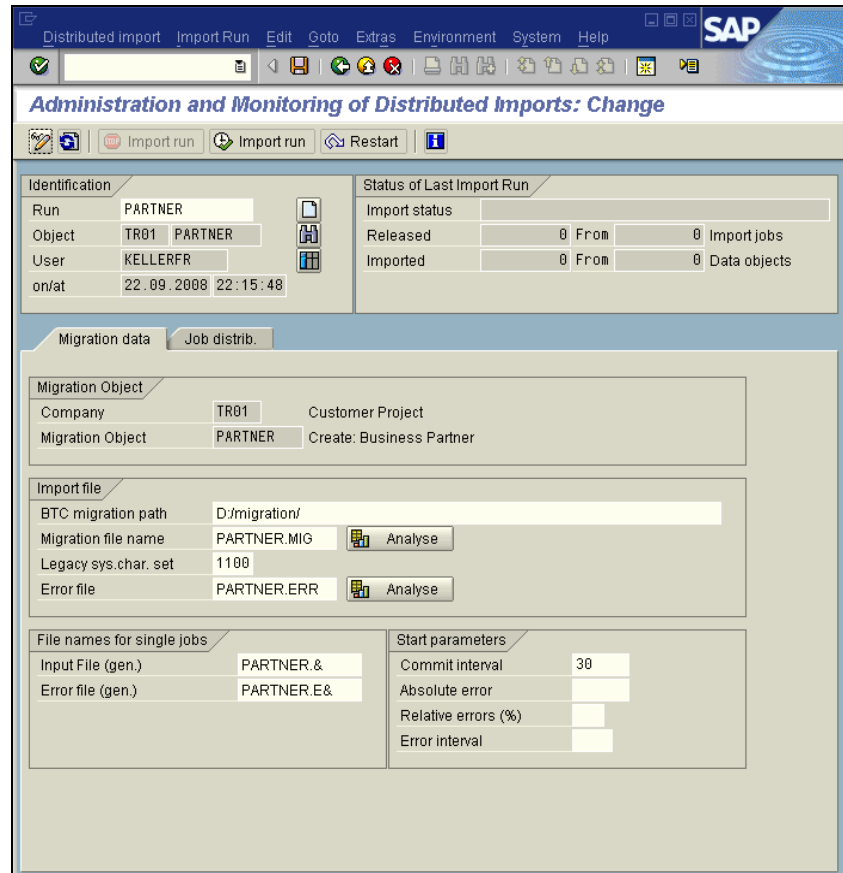


Figure 2-30: Data Flow Diagram of the Distributed Import

The **profile** of a distributed import defines and configures the characteristics of the data import. When creating a profile for a distributed import, it is important to define the name of the import file, the file size for the split files, and an initial distribution of the import runs on the application server. Figure 2-31 shows the administration screen of a distributed import of a newly defined profile.



The screenshot shows the SAP 'Administration and Monitoring of Distributed Imports: Change' screen. The interface includes a menu bar (Distributed import, Import Run, Edit, Goto, Extras, Environment, System, Help) and a toolbar with icons for various actions. The main content area is divided into several sections:

- Identification:** Fields for Run (PARTNER), Object (TR01 PARTNER), User (KELLERFR), and on/at (22.09.2008 22:15:48).
- Status of Last Import Run:** Fields for Import status, Released (0 From 0 Import jobs), and Imported (0 From 0 Data objects).
- Migration data / Job distrib. tabs:**
 - Migration Object:** Company (TR01), Customer Project, Migration Object (PARTNER), Create: Business Partner.
 - Import file:** BTC migration path (D:/migration/), Migration file name (PARTNER.MIG), Legacy sys.char. set (1100), Error file (PARTNER.ERR). Each file name has an 'Analyse' button.
 - File names for single jobs:** Input File (gen.) (PARTNER.&), Error file (gen.) (PARTNER.E&).
 - Start parameters:** Commit interval (30), Absolute error, Relative errors (%), Error interval.

Figure 2-31: Administration Screen of a Distributed Import with a New Profile

The names of the split files and the error files are defined using the placeholder **&**. The master job replaces the placeholder **&** with a consecutive three-digit number to create unique file names. The **Generic Suffix Length** parameter in the maintenance screen of the migration company is different and uses a four-digit number (for more information, see Figure 2-2). This parameter defines the maximum number of split files with a unique file name.

Figure 2-32 shows the screen for the definition of the job distribution. You can define for each available application server an individual number of jobs and change the job distribution at any time during an active data import. Utilize the system resources in this way to improve the performance of a data load. You can increase the system load (usage of system resources) in the database server and each of the application servers by scheduling more jobs, resolve a system overload of an application server or the database server by reducing the number of scheduled jobs, or redistributing the system load by adjusting the number of slave jobs on the available application servers. The master job monitors changes of the job distribution made by the user. It schedules more jobs, or stops scheduling jobs for a particular application server to adjust the active job distribution to the changed one.


Serv. name	Defined	Active	BGD	UPD	UP2
us4479_P7I_79	3	0	3	2	0
us4493_P7I_79	2	0	3	2	0

Figure 2-32: Definition of Job Distribution

The **Mass import file size** parameter is the final parameter required to complete the definition of the profile of the distributed import. This parameter defines the number of data objects (which are representing the data for one business object) in a split file. The final split file may have less data objects than the defined number.

You can start a distributed import immediately, or you can schedule it for an execution at a latter point. Figure 2-33 shows the screen where you start a distributed import.

Figure 2-33: Screen to Start a Distributed Import

You can stop an active distributed import in a controlled way by pushing the  button. The master job discontinues scheduling more slave jobs and waits for the active slave jobs to finish. A collection of the error file does not take place for a distributed import that you have stopped. You cannot restart a distributed import until the master job has finished.

After the first start of a distributed import for a profile, two more sub screens are available. Figure 2-34 shows the sub screen of the active or last active distributed data load of a distributed import. Links are available to display further detailed information.





Migration data		Job distrib.		Data for last import		Statistics for import runs	
Import run identification							
Migrator	KELLERFR	No. Repetitions	1				
Migration date	25.09.2008	<input type="checkbox"/> Restart					
Migration start time	14:40:11	 Migration Stats					
Statistical Data							
 No. of Import Files	20	Total data objects	2000	Obj			
Completely Processed	20	Imported	2000	Obj			
Currently Being Proc.	0	Incorrect	24	Obj			
Canceled	0	Already Migrated	0	Obj			
Previous Runtime	000:02:59	Hr	Throughput	0	Obj/Hr		
Data for master job							
Job name	REMIG_MASS_RUN_MASTER BTC	 Job Data					
Job number	14401000	 Job Log					

Figure 2-34: Data of the Last Data Load

Figure 2-35 shows the sub screen with a summary of the executed distributed data loads. For further details, double-click one of the records.

[illegible]

Figure 2-35: Summary of the Executed Distributed Data Loads

2.7.3 Group Import

You can group and execute several import runs with a distributed import in a chain. This function allows the execution of unattended migration test cycles (for example, overnight). To configure and execute distributed imports in groups, choose *Utilities* → *Schedule Import Jobs* → *Group Import*. Figure 2-36 shows the maintenance screen for the group import. You can mark each distributed import to be part of the next group import (**Active** indicator). The **Import Runs** sub screen shows, per import run, the history and other details of each distributed import when it is executed within the group.

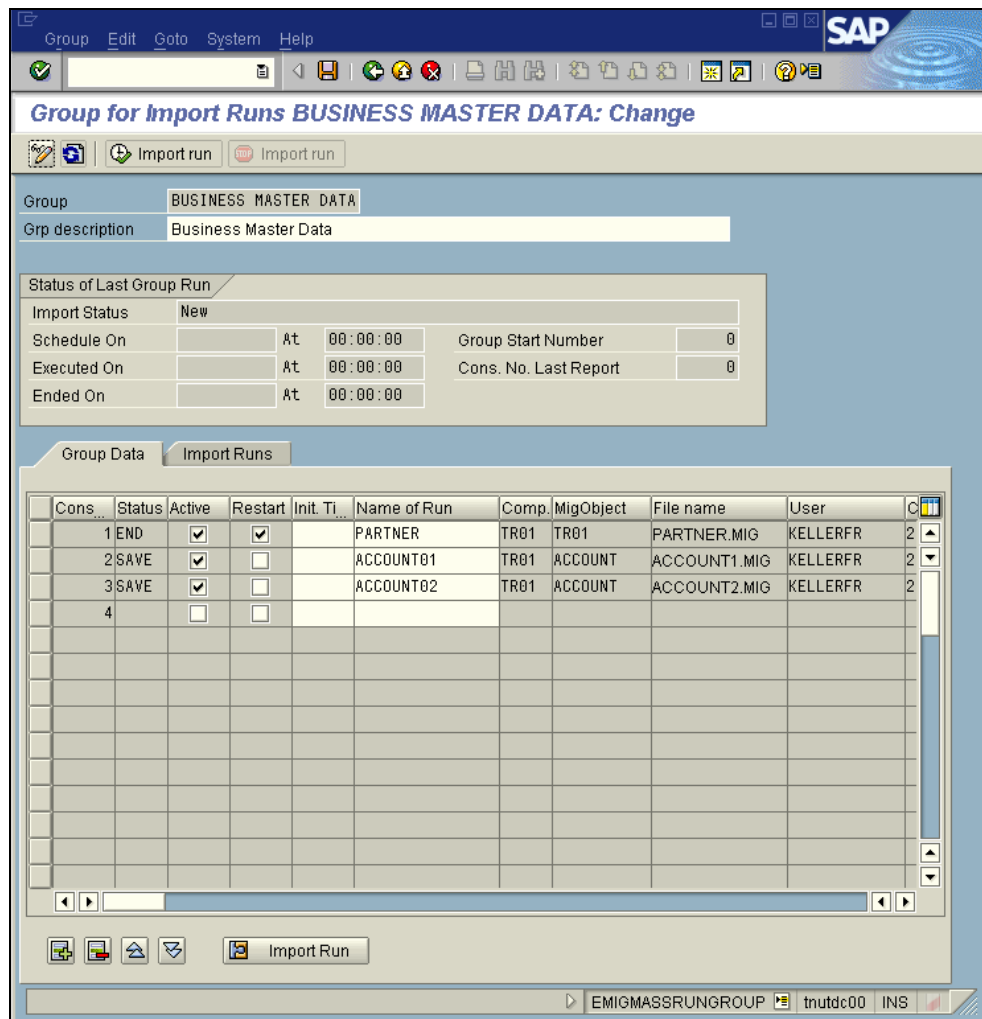


Figure 2-36: Group Import of Distributed Imports

The group import does not require a master job to monitor the scheduled distributed imports. The scheduled master job of the first distributed import receives the information that you have started it as part of a group import. Before the master jobs finishes, it schedules the next distributed import in the group list and passes the information about the group to the newly scheduled master job. You can stop the group import by pushing the **Stop Import Run** button. When you stop the execution of an active distributed import on the maintenance screen of the distributed import, this stops the distributed import and the group import from being continued.



You cannot develop complex process chains with the group import. However, you can include the execution of own programs in a group import. For more information, see chapter 7.12.6 *How to Execute Programs in a Group Import?*.

2.7.4 Alternative Job Distribution

The job distribution must be determined individually per migration object, based on the available system landscape and the business objects to be migrated. An alternative job distribution may be helpful if you have to process more than one import file per migration object, or if the job distribution is the same for various objects. Instead of defining a job distribution per distributed import in its profile, you define a variant of a job distribution once and link the profile of the respective distributed import to this variant. This variant is called **alternative job distribution**. This simplifies adjustments of the job distribution (for example,

if one of the application servers becomes unavailable). The alternative job distribution becomes active for a distributed import by entering the name of an alternative job distribution in the **Alt. Job distribution** field and marking the **Activate Alternative Job Distribution** field (for more information, see Figure 2-32).

You can define an alternative job distribution by entering a new name in the **Alt. Job distribution** field and then choosing the **Create** button or by choosing **Distributed Import → Job Distribution → Create**. Figure 2-37 shows the maintenance screen of an alternative job distribution.

Change Job Distribution

Job Distribution Edit Goto System Help

Identification

Alt. Job Distribution: PARTNER Status: Saved

Job Dist. Desc.: Alternative Job Distribution for the migration of PARTNER

For Migration Class: PARTNER Business Partner

Created on: 25.09.2008 Changed on:

Created by: KELLERFR Changed by:

Job Distrib.

Define distribution of import jobs

Serv. name	Defined	Active	BGD	UPD	UP2
us4479_P7I_79	2	0	3	2	0
us4493_P7I_79	1	0	3	2	0

Current Distribution

Work Processes

BGD: 6 UPD: 4 UP2: 0

Import Jobs

Defines: 3

P7I (1) (300) us4479 INS

Figure 2-37: Maintenance Screen for an Alternative Job Distribution

2.7.5 User-Exit in Master Job

You can control the execution of a distributed import with an event. The master job of a distributed import calls a customer-specific function module repeatedly:

- After the initializing phase but before the first file split; the `X_STATUS` parameter is set to `STRT`)
- When checking the status of the slave jobs; the `X_STATUS` parameter is set to `RUN`
- Before finishing; the `X_STATUS` parameter is set to `END`

If the distributed import is stopped either manually or automatically, the customer-specific function module is called once more with the `X_STATUS` parameter is set to `STOP`.

Figure 2-38 shows the `ISU_M_SAMPLE_DISTR_IMPORT` sample function module. You can use this as a basis for an own development. The only return parameter is the `Y_STATUS` parameter. You can stop the distributed import by returning the value `STOP` in this parameter. The calling program (`REMIG_MASS_RUN_MASTER` report) ignores any other value passed in this field. There are no programming restrictions available. The execution of a `COMMIT WORK` is permitted. For more information about how to send an e-mail to inform the migration team about the status of the data import, see chapter 9.5 *Function Module to Send a Status Email During Data Import*.

```

FUNCTION ISU_M_SAMPLE_DISTR_IMPORT.
** -----
** " Lokale Schnittstelle:
** " IMPORTING
** "   VALUE(X_STATUS) TYPE   EMG_STATUS OPTIONAL
** "   VALUE(X_RUNGROUP) TYPE EMG_RUNGROUP OPTIONAL
** "   VALUE(X_RUNGROUP_SEQ_NUMBER) TYPE EMG_SEQ_NUMBER OPTIONAL
** "   VALUE(X_TEMRUN) TYPE   TEMRUN OPTIONAL
** "   VALUE(X_TEMRUNMASTER) TYPE TEMRUNMASTER OPTIONAL
** "   VALUE(X_TEMSTATISTIK) TYPE TEMSTATISTIK OPTIONAL
** " EXPORTING
** "   VALUE(Y_STATUS) TYPE   EMG_STATUS
** " TABLES
** "   T_RUNDIST STRUCTURE TEMRUNDIST OPTIONAL
** "   T_RUNJOBS STRUCTURE TEMRUNJOBS OPTIONAL
** -----
ENDFUNCTION.

```

Figure 2-38: Sample for Custom-Specific Function Module

The custom-specific function module must be specified and made known to the distributed import with the `EVENT_DIST_IMPORT` control parameter of the IS Migration Workbench. A definition is possible globally, or you can restrict it to a specified migration company or migration object (for more information about how to define control parameters, see chapter 3.6 *Migration Control Parameter*).

2.8 Migration Lock Mechanism

Locks prevent data objects from being migrated if they cannot be created consistently (for example, data objects become chronologically incorrect due to a lack of a time slice), for a restart, or due to faulty data in previously processed data records. You must not confuse the lock mechanism in the IS Migration Workbench with the SAP system locks (transaction `SM12`). The locks in the IS Migration Workbench are only effective during an import run (within the actual work process). You can enable locking for business objects whose keys are processed by the *via KSM* field rule.

These parameters control the lock mechanism at field level:

- Check lock (business object is checked against the lock table)
- Lock object (business object is locked if data object cannot be processed)

Figure 2-39 shows the indicators on the field maintenance screen.

The screenshot shows the 'Initial' tab of the IS Migration Workbench. The 'Object Lock' section is highlighted with a red box, indicating that the 'Check lock' and 'Lock Object' options are enabled. The 'KSA control' section shows that 'Gen. KSM indic.' and 'Gen. Obj. Field' are not selected. The 'MigObject' list shows 'MigObject 1' with the value 'INSTLN' and the description 'Create Utility Installation'.

Figure 2-39: Control of Lock Mechanism on Field Level

Enable the lock mechanism on migration object level with the **ObjLockMechanism** indicator. Figure 3-15 shows the indicator on the **Migration Object Maintenance** screen.

The screenshot shows the 'Parameters' tab of the 'Migration Object Maintenance' screen. The 'ObjLock Mechanism' checkbox is checked and highlighted with a red box. The 'Import Function Mod.' field contains the value 'ZISU_M_IMPORT_TR01_INSTLNCHA'. The 'Specification for commit control' section shows 'Buffering allwd' checked and 'Commit interval' set to 1000. The 'Specification for cancellation ctrl' section shows 'Absolute error', 'Rel. error in %', and 'Error interval' fields.

Figure 2-40: Control of Lock Mechanism on Object Level

The load report maintains and manages the lock table during a data load. The load program adds the oldkey of the business object (currently processed by the field rule *via KSM*) together with the name of its top object to the process internal lock table if the *Lock Object* indicator is marked and if the data migration of the data object fails due to a problem. The load report checks if a processed business object is currently marked as locked. This check only takes place if the *Check Object* indicator is marked. In case of a restart, no locks are added to the lock table if a data object has already been migrated successfully.

The following figures explain the lock mechanism. A hyper migration object has been created containing the INST_MGMT (device installation) and DEVICEREL (device relationship) migration objects. The creation of the device relationship fails due to a data problem.

No lock mechanism becomes active in the first scenario shown in Figure 2-41. The deletion of the device relationship fails, while the device removal from the installation remains

possible. The problem that now occurs is that during the device removal, a time slice for the device is created. This makes a later migration of the device relationship impossible.

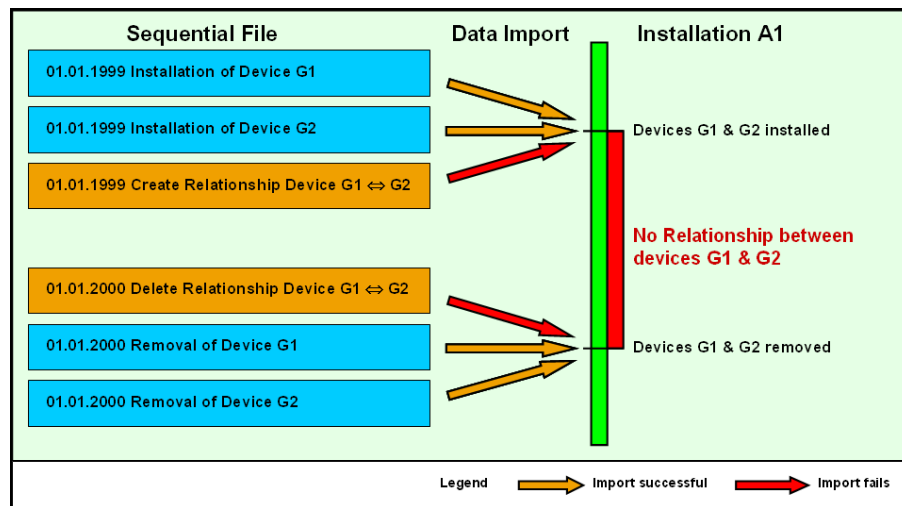


Figure 2-41: Example: Data Import Without Lock Mechanism

In the second scenario, the lock mechanism is activated for all fields with reference to devices in the migration objects INST_MGMT, DEVICEREL. It is also enabled on migration object level. If the creation of the device relationship fails, the application adds both devices to the process internal lock table. During the processing of the field rules for the deletion of the device relationship, the data object is rejected because both devices are marked as blocked in the lock table. This also happens during the processing of the device removals. The system does not remove the devices as they are already marked as blocked. Figure 2-42 shows the described process.

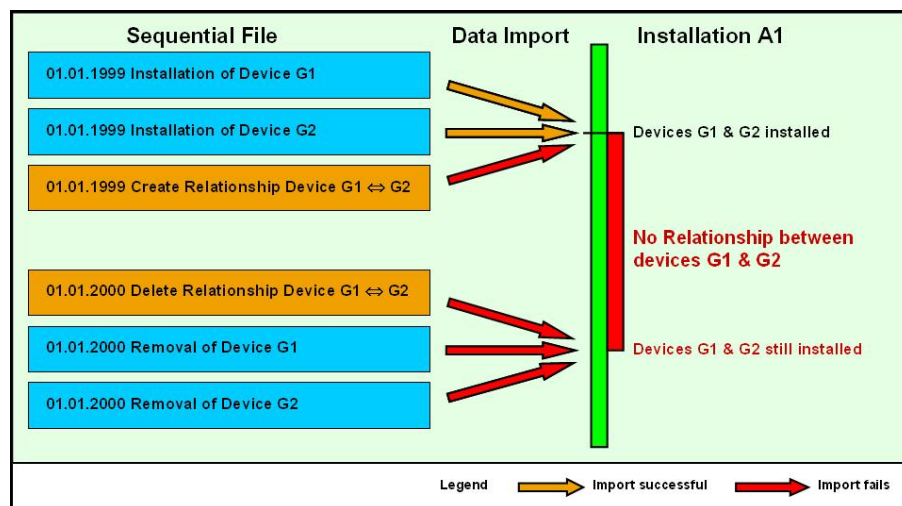


Figure 2-42: Example: Data Import with Lock Mechanism

You can migrate the device relationship, and its deletion and removal from the installation, at a later point in a restart. Before this can occur, ensure that the data in the import file for the creation of the device relationship is correct and you have resolved any problems.

The ISU_M_DOWNLOAD_LOCK_TABLE function module is available to save the internal lock table in a database table. You can use the ISU_M_UPLOAD_LOCK_TABLE function module to load the previously saved lock table into the process internal lock table. This may be required if an import file has been split into multiple files and the locks must be maintained across the sequentially processed import files. In this case, implement the

ISU_M_UPLOAD_LOCK_TABLE function module in an event at the generation point START. The ISU_M_DOWNLOAD_LOCK_TABLE function module must be named in an event at the generation point END (for more information, see chapter 4.3.3 *Code on Report* how to implement events).

2.9 Authorization

The ABAP authorization concept protects transactions, programs, and services in SAP systems from unauthorized access. The administrator assigns authorizations to the users. These authorizations determine which actions a user can execute in the SAP system when they have logged on. To access business objects or execute SAP transactions, a user requires corresponding authorizations.

The F_KKMIGRAT authorization object protects the IS Migration Workbench. In previous releases (including release 4.72), the E_MIGRATN authorization object (authorization class IS_U) has been used. Profiles can be designed with different values for the migration company, migration object, and for any action within the IS Migration Workbench. Additionally, the authorization for the EMIGALL transaction code of the IS Migration Workbench can be set accordingly.

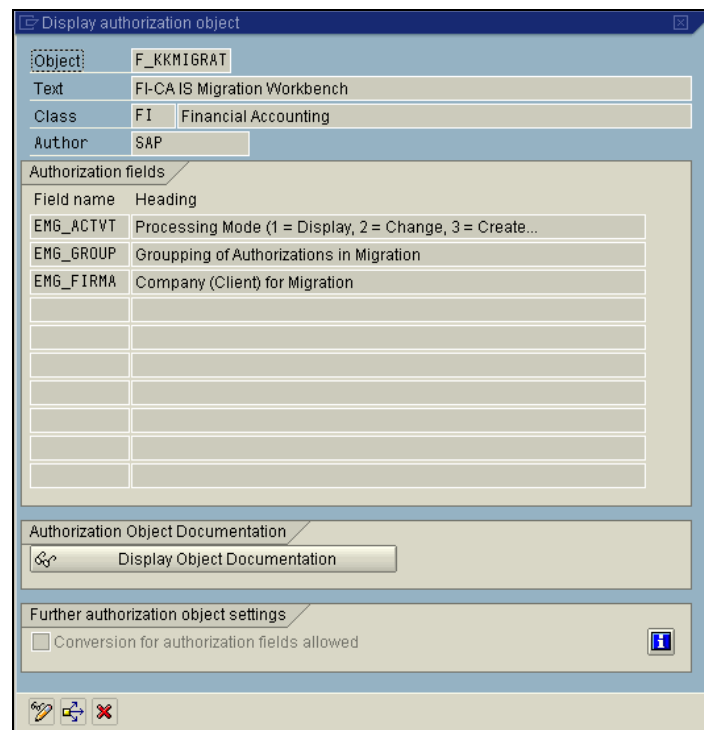


Figure 2-43 Authorization Object F_KKMIGRAT

In addition to the F_KKMIGRAT authorization object, the S_DEVELOP authorization object protects the creation and change of custom code in a load program and the upload of a migration object from a text file.

! It is important to understand that the F_KKMIGRAT authorization object controls only the execution of the function of the IS Migration Workbench. To create and change data in the SAP system, a migration user must have authorization to do so. This is because the migration user executes the data migration and calls the necessary application. The application checks whether the migration user requires allocation of further authorizations.

Use the same authorization profile for the migration user for all data migration cycles. Many projects experience serious problems due to the allocation of different authorization profiles

in dependency into which system data was migrated. For a data migration into the migration system the profile `SAP_ALL` (full authorization) was allocated to the migration user, but only a reduced profile has been approved for the data migration user for a migration into the production system. Problems can now occur in the production system due to lack of authorization for a creation or change of business objects during data migration.

By treating a data migration cycle into a test system as if it were a data migration into the production system, it is possible to identify authorization problems before going live. Usually, unlimited authorization (profile `SAP_ALL`) for a migration user cannot be granted. However, the migration user must have sufficient authorization to do a basic system monitoring during any data load.

We recommend that you use a generic user name for the migration users during the migration test cycles, as well as when you go live. This is because the user name of the migration user is stored in most migrated data objects and transactions in the SAP system. A generic user name (for example, `MIGRATION`) indicates that the used data has been migrated.




We recommend that you restrict access to the IS Migration Workbench after going live. To do this, remove at least the authorization for changes to migration objects and for an execution of a data import. Otherwise, an unauthorized execution of ABAP code may occur in the production system, due to the IS Migration Workbench generating an executable ABAP report based, for example, on events.

You can use the blocking status of a migration object to block, for example, the field maintenance or the data import. In this way, you can prevent unwanted changes to the customer structure and inadvertently starting a migration report without completed Customizing. Figure 2-44 shows the available blocking status. Set the blocking status of a migration object on the migration object maintenance screen.


Blocking Status	Description
000	No block active
001	Object maintenance blocked
002	Structure maintenance blocked
003	Field maintenance blocked
010	Report generation blocked
011	Report generation and object maintenance blocked
012	Report generation and structure maintenance blocked
013	Report generation and field maintenance blocked
100	Data import blocked
101	Data import and object maintenance blocked
102	Data import and structure maintenance blocked
103	Data import and field maintenance blocked
110	Data import and report generation blocked
111	Data import, report generation, and object maintenance blocked
112	Data import, report generation, and structure maintenance blocked
113	Data import, report generation, and field maintenance blocked

Figure 2-44 Blocking Status for Migration Objects

 You must not confuse the blocking status of a migration object in the IS Migration Workbench with the ABAP authorization concept. If you are authorized to maintain a migration object, you can change the blocking status of a migration object.

2.10 Change Requests

When objects in the ABAP Workbench or Customizing are created or modified, a window appears asking you to assign the object to an existing change request (own requests), or to create a new change request (create request). The aim of the development work is to introduce new or improved functions into the production system. The created change requests are transported into subsequent SAP systems and finally into the production system.

 The IS Migration Workbench does not automatically record changes to the migration Customizing. You can save the Customizing of migration objects in text files, or in change requests, to transport migration specific Customizing from one SAP system to another SAP system.

2.10.1 Transport with Change Requests

The transport of a change request uses the Change and Transport System (CTS). Save the various migration Customizing in change requests. Figure 2-45 shows a table with the access paths in the IS Migration Workbench for all Customizing objects.


Customizing	Access Path
Migration Object	<i>Utilities → Transport Migration Object → Transport via Transport Request</i>
Fixed Value Object	Transport only together with migration object
Conversion Object	Transport only together with migration object
Distributed Import	<i>Utilities → Schedule Import Jobs → Distributed Import → Extras → Transport Request</i>
Control Parameter	<i>IS Migration → Settings → Customizing Settings → Button Create Transport Request</i>

Figure 2-45 Access Paths to Save Migration Customizing in Change Requests

All objects holding migration Customizing are defined as logical transport objects or as table transport objects (transaction SOBJ). The TEMO logical transport object is defined with an after import method ISU_M_TEMO_AFTER_IMP. This method forces a (re-)generation of the load report for the related migration object after the import of a transport request.

Customizing	Object	Object Type
Migration Object	TEMO	L = Logical Trsp. Object
Fixed Value Object	TEMF	L = Logical Trsp. Object
Conversion Object	TEMC	L = Logical Trsp. Object
Distributed Import	TEMR	L = Logical Trsp. Object
Control Parameter	TEMCONTROLC	S = Table

Figure 2-46 Transport Objects for Customizing Objects

 The after import method ISU_M_TEMO_AFTER_IMP checks during the import whether the related migration company already exists. If the migration company does not exist, the method creates the migration company with default values. A migration company cannot

be transported. Therefore, you must either create your migration company before importing the transport request or adjust the migration company created by the after import method `ISU_M_TEMO_AFTER_IMP`.



To ensure the consistency of the migration Customizing in the target system, the IS Migration Workbench transports the Customizing of always all referenced migration objects Always together with the Customizing of the hyper migration object (for more information, see chapter 3.5 *Hyper Object*).

2.10.2 Transport of Migration Objects in Text Files

This transport process allows the transportation of the Customizing of migration objects via text files by using the PC download and upload functions. To start the transport, choose *Utilities → Transport Migration Object → Transport via local file*. To complete the transport of the migration object Customizing, add the defined fixed values and conversion values. The IS Migration Workbench compares the release information saved in the text file with the release of the SAP system. If the releases are not the same, the upload of a migration Customizing may cause inconsistencies in migration objects. In this case, the SAP system raises a warning message. You should prefer the transport of Customizing in transport requests over transports in text files.

3 Migration Object

3.1 Definition

A migration object is a logical unit that combines the functional and technical aspects for a migration of data into the SAP system. The basis for the technical definition of a migration object is an ABAP function module and its parameter interface. The parameter interface of the function module defines the automation data of the migration object. A function module used in a migration object is called a service function module; its interface is called automation data structure. The service function module passes the data through the automation data structure to the application, which creates the requested business master data or transaction data. Each migration object allows the data migration of a specific business object as business master data or transaction data (for example, the master data object business partner).

Figure 3-1 shows the object maintenance screen of the IS Migration Workbench with the definition of the PARTNER migration object that allows the migration of business partners.

The screenshot displays the 'Object Data' tab of the migration object maintenance screen. The top section contains fields for 'Company' (CUST), 'Migration Company Customer', 'Mig. object' (PARTNER), 'Mig. class' (PARTNER), 'Mig. obj. text' (Create: Business Partner), 'Mig. obj. abbr.' (PAR), 'Blocking Status' (100), and 'Data import blocked'. Below this, the 'Object Data' tab is active, showing the 'Service module' (ISU_M_PARTNER_CREATE_DARK) with an 'Interface' button, 'Auto struc.type' (ISUMI_PARTNER_AUTO), 'Return structure' (NEW_PAR), and 'Return field' (NEW_PAR-PARTNER). There are also checkboxes for 'Internal Table' and 'Structure type' (checked), and a field for 'BUT000'.

Figure 3-1: Definition of Migration Object PARTNER

The PARTNER migration object uses the ISU_M_PARTNER_CREATE_DARK function module as the service function module and requires the automation data structure ISUMI_PARTNER_AUTO as defined in the **Auto struc. Type** field. The service function module returns the SAP key (newkey) of a created business partner in the NEW_PAR-PARTNER field as defined in the **Return field** field.

The automation data structure of a migration object has a hierarchical structure. The complexity of the structure depends on the complexity of the business object to be migrated and the parameter interface of the service function module. Figure 3-2 shows the most important structures of the automation data structure of the PARTNER migration object.

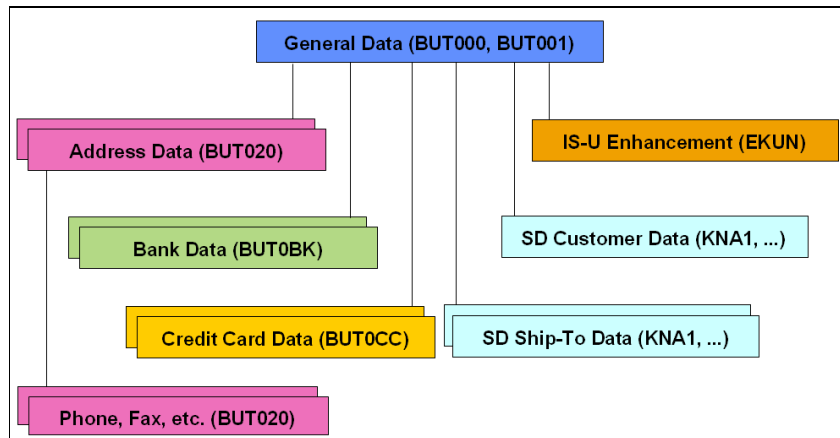


Figure 3-2 Hierarchical Automation Data Structure of Migration Object PARTNER

In Figure 3-3, the same PARTNER migration object is shown in the main screen of the IS Migration Workbench. It displays information regarding the applied Customizing of the migration object in structure and field level. The first column is reserved for the name of the migration objects, the second column shows all structures as defined in the automation data, and the third column shows all the fields of the marked structure. By default, only the structures and fields marked with the button are shown. To show all available structures choose *Automatic Structure* → *Display List* → *All*. To show all fields for a selected structure, choose *Field* → *Display List* → *All*.

The workbench status (WB status) area shows the selected migration object with the selected structure. The symbol indicates that further migration specific documentation is available. Choose *Migration Object* → *Display Object documentation* to display the migration object specific documentation, or *Automatic Structure* → *Display Structure Documentation* for structure specific documentation.

WB status		
Object: PARTNER		Create: Business Partner
Struct.: INIT		Header Data for Business Partner
MigObject	Auto. Struct.	Fld
PARTNER	INIT	
	EKUN	BPKIND 3
	BUT000	BU_GROUP 4
	BUTICOM	BU_RLTYP 4
	BUT0BK	BU_TYPE 4
	BUT020	MUSTER_KUN 2
	BUT021	PARTNER 3
	BUT0CC	
	SHIPTO	
	TAXNUM	
	ECCARD	
	ECCARDH	
	BUT0ID	
	BUT0IS	
	KNB5	
	KNB1	
	KNA1	
	KNVV	
	LFA1	
	LFB1	

Figure 3-3 Automation Structure of the PARTNER Migration Object

3.2 Migration Class

A migration object is based on a business object. The migration class uniquely links the migration object with an object category in the Business Object Repository (BOR). This information is used to display the data objects that belong to the newkeys saved in KSM. With this, for example, the PARTNER migration object can be given an own name when it is copied from the SAP migration company to the project migration company. The IS Migration Workbench recognizes that the migration object is to migrate business partners. The IS Migration Workbench uses the migration class in various activities, such as, displaying a migrated business object from the import file editor.

Choose IS Migration → *Settings* → *Maintain Migration Class* to display the defined migration classes. There are special migration classes available like BDC (for more information, see chapter 3.4.1 *Generation of a BDC Migration Object*), BAPI (for more information, see chapter 3.4.2 *Generation of a BAPI Migration Object*), and HYPER (for more information, see chapter 3.5 *Hyper Object*). Figure 3-4 shows the definition of the PARTNER migration class. The BOR object type for the business partner is BUS1006.

Migration class:	PARTNER
IS-U migration: maintenance view for migration class	
Class description	Business Partner
Function Module	
Object Type	BUS1006

Figure 3-4 Definition of Migration Class PARTNER

3.3 Pre-Configured Migration Objects

SAP delivers preconfigured migration objects in the SAP migration company. Depending on the activated SAP solutions, the migration objects that are not relevant to the active SAP solution are hidden. To display the delivered migration objects, choose *Migration Object* → *Display List* → *All Industries on*.

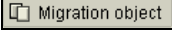
You can display a list of all available migration objects with a link to the documentation for each of the migration objects. Choose *IS Migration* → *Information System* → *List of migration objects*. Read the documentation of any migration object before you change the Customizing.

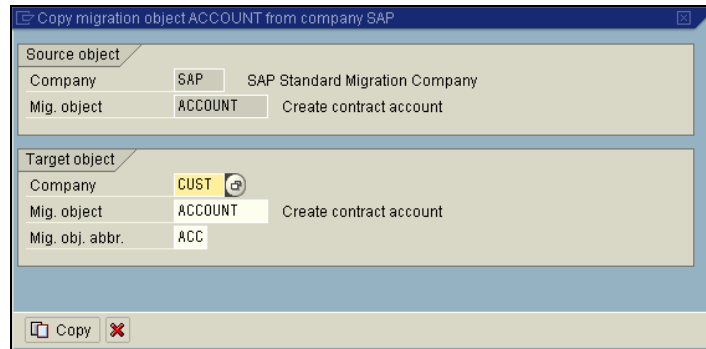
Avoid a change of any migration object in the SAP migration company. The application displays the following warning message when you try to change the Customizing for a migration object:

You wanted to a process standard migration object <name of migration object>
Only make changes to standard SAP migration companies and migration objects in emergencies since they can be overwritten with each upgrade or SP.
Only carry out changes in your own migration company

Figure 3-5: Warning when Changing a Migration Object in Migration Company SAP

Only change the migration Customizing in an own migration company. Create migration objects in the own migration company as a copy of a delivered migration object in the SAP migration company. You can do this in two ways: either copy the migration objects one by one into the own migration company, or select multiple migration objects for copying. Figure 3-6 shows how to copy a migration object from SAP migration company. To copy more than one migration object at a time, choose *Utilities* → *Copy Migration Objects*.

1. Change to the migration company where the migration object is located and select the migration object that you want to copy.
2. Choose *Migration Object* → *Copy*, or push the  button
3. Enter or change the name of the migration company into which you are copying the migration object. It might be necessary to change the name of migration object and the abbreviation as this information must be unique in each migration company.



4. Choose the *Copy* button.

Figure 3-6: Procedure to Copy a Migration Object

After you copied a migration object from migration company SAP to your migration company, the blocking status of the copied migration object is set to *100 Data Import blocked*. This is to avoid a usage of your migration object without having the adjusted the Customizing for your new migration object.

3.3.1 Migration Objects for all Solutions

The following shows all of the migration objects delivered in the SAP migration company that you can use for a data migration for all released SAP solutions. For more information, see the documentation of the migration object.

3.3.2 Migration Objects for Regional Structure

- **ADRPSTCODE (Create Postal Code for Postal Regional Structure)**
Use the ADRPSTCODE migration object to migrate regional structure data for postal codes.
You can migrate postal codes independently of other migration objects.
- **ADRCITY (Create City for Postal Regional Structure)**
Use the ADRCITY migration object is used for migrating regional structure data for cities.
You cannot migrate any city data until you have migrated the postal codes (ADRPSTCODE migration object).
- **ADRSTREET (Create Street for Postal Regional Structure)**
Use the ADRSTREET migration object to migrate regional structure data for streets and street sections.
A requirement for a migration of a street is the successful migration of a city (ADRCITY migration object).
- **REGPOLIT (Political Regional Structure)**
Use the REGPOLIT migration object to migrate data from the political regional structure.
A requirement for a migration of the political regional structure is the definition of the country-related hierarchy for modeling your political regional structure. You can execute the migration for the corresponding country at the same time as other import runs.

3.3.3 Migration Objects for Business Master Data

○ **PARTNER (Create Business Partner)**

The PARTNER migration object contains the fields for a master data object business partner. The migration object contains fields for the main business partner and solution specific data for the business partner. You can create a customer by specifying a reference in the Sales and Distribution (SD) application.

Migration of data for the business partner does not depend on other migration objects except in the creation of delivery addresses (SHIPTO structure), which requires the successful migration of the related connection objects (CONNOBJ migration object).

○ **PARTNERCHA (Change Business Partner)**

The PARTNERCHA migration object contains the fields for changing a business partner. The migration object contains the data for the business partner and SAP for Utilities data for the business partner, such as, the migration of the following:

- New addresses or changes to existing addresses
- New bank details or changes to bank details
- New credit card data or changes to credit card data
- Supplement newly included fields to the business partner as part of a customer enhancement (for example, CI_EKUN structure)
- Generation of SD customers for existing business partners

A requirement for the migration of a business partner change is the successful migration of the corresponding business partner (PARTNER migration object).

○ **PART_REL (Change Business Partner)**

Use the PART_REL migration object to create business partner relationships.

A requirement for the migration of a business partner relationship is the successful migration of the corresponding business partners (PARTNER migration object).

○ **ACCOUNT (Create Contract Account)**

The ACCOUNT migration object contains the fields for the master data object business contract account.

A requirement for the migration of a contract account is the successful migration of the corresponding business partner (PARTNER migration object).

○ **ACCOUNTCHA (Change Contract Account)**

Use the ACCOUNTCHA migration object to migrate changes to contract accounts.

A requirement for the migration of contract account changes is the successful migration of the corresponding contract account (ACCOUNT migration object).

○ **ACC_NOTE (Create Notes for Contract Account)**

Use the ACC_NOTE migration object to migrate the notes to the contract accounts.

The configuration delivered with this object is configured exclusively for migrating notes to contract accounts. You cannot use this migration object as a template for migration object to load notes for other business objects, other than notes for contract accounts. To do that, use the BCONT_NOTE migration object.

A requirement for the migration of a contract account note is the successful migration of the corresponding contract account (ACCOUNT migration object).

3.3.4 Migration Objects for Financial Data

Several migration objects have been defined in the IS Migration Workbench for migrating data for the contract accounts receivable and payable. Use these migration objects to implement a large number of possible scenarios.

Examples of these scenarios are as follows:

- Use the DOCUMENT migration object to migrate the balance of every contract account in the legacy system as a receivable or a credit memo as well as any open item.
You can also use the DOCUMENT migration object with PAYMENT migration object to migrate a payment history. The open items are migrated with their original amount using DOCUMENT migration object. Then, you can partially clear the open items by a payment with the PAYMENT migration object.
- Special migration objects are available for further financial documents, such as, cash security deposits, payments for them, and the calculation of their interest (SECURITY, PAYMENT, INTCASHDEP migration objects). The migration of cash security deposits, including the requests, payments, and interest calculations of a credit, is a process performed in several steps. They are as follows:
 1. Create of a cash security deposit with the SECURITY migration object. You can classify the security deposit as a cash security deposit using the *NON_CASH* field. During the creation of a cash security deposit, the system posts a request.
 2. Migrate of cash a security deposit with PAYMENT migration object. Define a field rule to determine the document number from the security deposit.
 3. Migrate the interest information with INTCASHDEP migration object. This migration object generates the necessary interest document for credit originating from cash security deposits.

The migration objects suitable for a migration of financial data are as follows:

- **BANK (Create Bank Address)**
Use the BANK migration object to create bank master data. The SAP standard system contains many programs that can copy bank master data from certain sources. For more information on the data sources, providers, and countries supported by these programs, see the Implementation Management Guide (IMG) under *Bank Directory*. Use the BANK migration object if there is no information in the IMG on how to migrate bank data for your country. You can also use the BANK migration object if your bank data has a different source than those specified in the IMG.
Bank master data is migrated independently of any other migration object.
- **DOCUMENT (Create FI-CA Document as an Open Item)**
The DOCUMENT migration object contains the data for (FI-CA) documents. You can migrate open items for the FI-CA component (contract accounts receivable and payable) using this migration object.
A requirement for a migration of an open item is the successful migration of the corresponding business partner (PARTNER migration object) and contract account (ACCOUNT migration object). For the migration of data for the solution SAP for Utilities, additionally contracts (MOVE_IN migration object) have to be migrated successfully.
- **PAYMENT (Create Payment on Open Items)**
The PAYMENT migration object contains all the fields for migrating payment documents to (FI-CA) documents.
A requirement for a migration of a payment is the successful migration of the corresponding business partner (PARTNER migration object), contract accounts

(ACCOUNT migration object), FI-CA documents (DOCUMENT migration object), and cash security deposit (SECURITY migration object).

- **DOC_DOUBT (Migration of Individual Value Adjustment/Doubtful Entry Supply)**

The DOC_DOUBT migration object includes the fields to migrating doubtful entries or individual value adjustment documents (FI-CA). Use this migration object to create trigger records for open items for doubtful entries or individual value adjustments that you have migrated with migration object DOCUMENT. With the mass activity FPRV, the items in the general ledger (doubtful entries) are written-off or the expenses are posted to the general ledger for individual value adjustment.

A requirement for a migration of a doubtful entry or individual value is the successful migration of the corresponding business partner (PARTNER migration object), contract accounts (ACCOUNT migration object), and FI-CA documents (DOCUMENT migration object).

- **DOC_INT (Allocation of the Last Interest Dates for Open Items)**

The DOC_INT migration object contains the fields required to migrating interest of an open item. Use this migration object to migrate the date of the last interest calculation for open items already migrated with the DOCUMENT migration object.

A requirement for a migration of an interest document is the successful migration of the corresponding business partner (PARTNER migration object), contract accounts (ACCOUNT migration object), and FI-CA documents (DOCUMENT migration object).

- **REQUEST (Create Request)**

The REQUEST migration object incorporates data for requests in contract accounts receivable and payable. Use this migration object to create requests and standing requests. In the public sector, contract accounts receivable and payable can also create general requests.

A requirement for a migration of a request is the successful migration of the corresponding business partner (PARTNER migration object), contract account (ACCOUNT migration object), and contract (the migration objects from the relevant industry component has to be used to create contracts)

- **INSTPLAN (Create Installment Plan)**

The INSTPLAN migration object contains the fields for migrating installment plans.

Use this migration object to migrating installment plans for open items of the contract accounts receivable and payable component (FI-CA) that have already been migrated with the DOCUMENT migration object. This enables you to add one or more open items to the installment plan. The individual due dates can be generated in accordance with the situation in the legacy system (date and amount of the due date). In the data migration process, no proposal comes from dialog customizing for the installment plan.

A requirement for a migration of an installment plan is the successful migration of the corresponding business partner (PARTNER migration object), contract account (ACCOUNT migration object), and FI-CA documents (DOCUMENT migration object).

- **SECURITY (Create Security Deposit)**

The SECURITY migration object contains the fields for the security deposit master data object. When a cash security deposit is migrated, a request is generated in accordance with the Customizing settings

A requirement for a migration of a security deposit is the successful migration of the corresponding business partners (PARTNER migration object) and contract accounts (ACCOUNT migration object).

- **INTCASHDEP (Create Interest Document for Cash Security Deposit)**

The INTCASHDEP migration object contains the fields to migrate interest documents for payments on requests of cash security deposits. It can only calculate interest for single payment documents.

A requirement for a migration of interests for security deposits is the successful migration of the corresponding business partner (PARTNER migration object), contract account (ACCOUNT migration object), security deposit (SECURITY migration object), and payment (PAYMENT migration object).

- **DUNNING(Create Dunning Items and Creditworthiness)**

The DUNNING migration object contains the fields for migrating dunning data for FI-CA documents and creditworthiness data for business partners. Use this migration object to migrate dunning data for the open items that have already been migrated with DOCUMENT migration object. You can set up a dunning history.

A requirement for a migration of dunning history is the successful migration of the corresponding business partner (PARTNER migration object), contract account (ACCOUNT migration object), and FI-CA document (DOCUMENT migration object).

3.3.5 Migration Objects for Miscellaneous Data

- **BCONTACT (Create Customer Contact)**

The BCONTACT migration object contains the fields to migrate customer contacts.

A requirement for a migration of a contact is the successful migration of the corresponding business partner PARTNER (migration object).

- **BCONT_NOTE (Create Note)**

The BCONT_NOTE migration object contains the fields to migrate notes for a contact. The migration object is configured to migrate notes for customer contacts. If notes need to be migrated to other business objects (for example, business partners or premises), you must copy the migration object and make the corresponding adjustments.

A requirement for a migration of a note is the successful migration of the corresponding business partner (PARTNER migration object) and customer contact (BCONTACT migration object).

- **OFFICEDOC (Create and Send SAP Office Document)**

The OFFICEDOC migration object contains the fields to create SAP Office documents during migration.

A requirement for a migration of an office document is the successful migration of the corresponding business objects.

3.3.6 Migration Objects Specific to SAP for Utilities

The following migration objects are specific to the industry solution SAP for Utilities.

3.3.6.1 Migration Objects for Regional Structure

- **ADRCITYISU (Data for Cities for SAP for Utilities)**

Use the ADRCITYISU migration object to enhance the regional structure data for cities to include data for SAP for Utilities (such as meter reading units, franchise contracts, and company codes). You can only change SAP for Utilities specific city data here. If necessary, it is possible to delete existing SAP for Utilities data (such as meter reading units).

A requirement for a migration of SAP for Utilities specific city data is the successful migration of the corresponding city (ADRCITY migration object).

- **ADRSTRTISU (Data for Streets for SAP for Utilities)**

Use the ADRSTRTISU migration object to enhance the regional structure data for cities to include SAP for Utilities data (such as meter reading units, franchise contracts, and company codes). You can change only SAP for Utilities specific city data changed here. If necessary, it is possible to delete already existing SAP for Utilities data (such as meter reading units).

A requirement for a migration of SAP for Utilities specific street data is the successful migration of the corresponding street (ADRSTREET migration object).

3.3.6.2 Migration Objects for Technical Master Data

- **CONNECTION (Create Connection)**

The CONNECTION migration object contains the fields for the technical object connection. This migration object includes both the equipment data from the Plant Maintenance (PM) component and the configuration data. A configurable material is allocated to the equipment. You can then allocate values to the characteristics of the equipment.

Connections are migrated independently of other migration objects.

- **CONNOBJ (Create Connection Object)**

The CONNOBJ migration object contains the fields for the connection object. A functional location is created during creation of a connection object. Using this object, you can create SAP for Utilities data that is identical to the dialog transaction. You can change the data of the functional location created alongside SAP for Utilities data using the FUNCLOCCHA migration object.

Connection objects are migrated independently of other migration objects.

- **DEVGRP (Device Composition: Create, Undo, or Change Device Group)**

Use the DEVGRP migration object to create, terminate, and change the devices used in device groups. Install devices belonging to a device group in the same device location.

A requirement for a migration of a device group is the successful migration of the corresponding devices (DEVICE migration object).

- **DEVICE (Create Device or Equipment)**

The DEVICE migration object contains the fields for the technical object equipment. The migration object contains the equipment data of the plant maintenance (PM) component and the data for SAP for Utilities device. It is also possible to create classification data.

Devices are migrated independently of other migration objects.

- **DEVICEMOD (Modify Device)**

The DEVICEMOD migration object contains the fields to modify a device. It is possible to modify installed and devices which are not installed.

A requirement for a migration of a device modification is the successful migration of the corresponding devices (DEVICE migration object).

- **DEVINFOREC (Create Device Info Record)**

The DEVINFOREC migration object contains the fields for creating a device info record.

Device info records are migrated independently of other objects.

- **FUNCLOC (Create Functional Location)**

The FUNCLOC migration object contains the fields for creating the PM Master Data object functional location. It is not possible to create any connection objects or device locations with this object. You must use the appropriate CONNOBJ and DEVLOC migration objects.

Functional locations are migrated independently of other objects.

- **FUNCLOCCHA (Change Functional Location)**

The FUNCLOCCHA migration object contains the fields for changing the PM master data object functional location. With this migration object, you can change data of the functional locations that were created with the migration objects CONNOBJ (create connection object), DEVLOC (create device location), or FUNCLOC (create functional location). You can create address data for a device location. A change of address data is not possible.

A requirement for a migration of a functional location change is the successful migration of the corresponding function location (FUNCLOC migration object).

- **INSTLN (Create Utility Installation)**

The INSTLN migration object contains the fields for creating master data object installation for SAP for Utilities. A point of delivery is created when an installation is created. If *POD* structure is not used, or if no external relationship is transferred in the *POD* structure, then an internally numbered point of delivery with category **deregulation point of delivery** is created. You can create installation facts when you are creating the installation but we recommend that you migrate installation facts with the FACTS migration object.

A requirement for a migration of an installation is the successful migration of the corresponding premise (PREMISE migration object).

- **INSTLNCHA (Change Utility Installation)**

The INSTLNCHNA migration object contains the fields to change the installation of master data objects for SAP for Utilities. You must import changes of an installation in chronological order. We recommend that you use the FACTS migration object to migrate changes to installation facts.

A requirement for a migration of an installation change is the successful migration of the corresponding installation (INSTLN migration object).

- **FACTS (Create Individual Facts for Utility Installation)**

The FACTS migration object contains the fields for the creation or change of installation facts of the SAP for Utilities master data object installation.

A requirement for a migration of a fact is the successful migration of the corresponding installation (INSTLN migration object).

- **REFVALUES (Create Reference Values)**

The REFVALUES migration object contains the fields to create or change of reference values.

A requirement for a migration of a reference value is the successful migration of the corresponding installation (INSTLN migration object).

- **LOT (Sample Lot)**

The LOT migration object contains the fields required for migrating sample lots. This migration object creates only the SAMPLE LOT data object. It does not create the link between the lots and the devices.

A requirement for a migration of a lot is the successful migration of the corresponding devices (migration object DEVICE).

- **LOTFINAL (Completion Action: Sample Lot)**

The LOTFINAL migration object contains the fields to migrate completion data for creating sample lots. Using this function, you can assemble the lot devices that correspond to an existing lot (migrated with the LOT migration object) and enter the date specifications from the legacy system in the sample lot.

A requirement for a migration of a lot completion is the successful migration of the corresponding lot (migration object LOT).

- **MOVE_IN (Create Move-In or Utility Contract)**

The MOVE_IN migration object contains the fields to create active contracts. In the move-in process, one or more installations are allocated to a contract account and to a business partner. The move-in date is set as the start date for the new billing period.

A requirement for a migration of a contract is the successful migration of the corresponding contract account (ACCOUNT migration object) and installation (INSTLN migration object).

- **MOVE_OUT (Create Move-Out for Contract)**

The MOVE_OUT migration object contains the fields to create a move-out. One or more contracts are terminated during this procedure. It is sufficient to supply the move-out date and the contract itself. You can stop a budget-billing plan that is in operation at the same time. Possible existing move-out meter readings must be migrated separately (METERREAD migration object). When you process the move-out, the system automatically generates an order for a final billing.

A requirement for a migration of a move-out is the successful migration of the corresponding contract (MOVE_IN migration object).

- **MOVE_IN_H (Create Historical Utility Contract)**

The MOVE_IN_H migration object contains the fields for an integrated move-in and move-out transaction for historical contracts. With this migration object, an inactive contract and a billing trigger for this contract are created.

A requirement for a migration of a contract is the successful migration of the corresponding contract account (ACCOUNT migration object) and installation (INSTLN migration object).

- **NOTE_CON (Create Notes to Field Service for Connection Object)**

The NOTE_CON migration object contains the fields for creating static notes for a connection object. The field service uses these notes.

A requirement for a migration of a note for a connection object is the successful migration of the corresponding connection object (CONNOBJ migration object).

- **NOTE_DLC (Create Notes to Field Service for Device Location)**

The NOTE_DLC migration object contains the fields for creating static notes for a device location. The field service uses these notes.

A requirement for a migration of a note for a connection object is the successful migration of the corresponding device location (DEVLOC migration object).

- **POD (Point of Delivery)**

The migration object POD contains the fields for creating the master data object point of delivery (POD). You can allocate the point of delivery to one or more installations (possibly time-dependent) simultaneously. The allocation can be done separated by deregulation aspects (installation allocation) and technical aspects (allocation to installations, devices, or registers).

A requirement for a migration of a PoD might be the successful migration of the corresponding installation (INSTLN migration object) and devices (DEVICE migration object).

- **PODCHANGE (Change Point of Delivery)**

The PODCHANGE migration object contains the fields for changing the master data object point of delivery (POD). You can allocate a deregulated point of delivery to one or more installations (time-dependent).

A requirement for a migration of a PoD change is the successful migration of the corresponding PoD (POD migration object).

- **PODSERVICE (Create Point of Delivery Service)**
The PODSERVICE migration object contains the fields for allocating a point of delivery service to a PoD.
A requirement for a migration of a PoD service is the successful migration of the corresponding PoD (POD migration object).
- **PREMISE (Create Premise)**
The PREMISE migration object contains all the fields for creating the master data object premise.
A requirement for a migration of a premise is the successful migration of the corresponding connection object (CONNOBJ migration object).
- **PROPERTY (Create Owner Assignment)**
The PROPERTY migration object contains the fields for allocating a property. You allocate a connection object, a premise, or an installation to an owner (business partner). You can use the X_PRORATE transfer parameter of the ISU_S_PROPERTY_CREATE service function module to create multiple time slices to allocate different owners to a connection object and the allocated premise.
You can only migrate the data for the property assignments after the successful migration of the following migration objects:
A requirement for a migration of a property allocation is the successful migration of the corresponding business partner (PARTNER) and installation (INSTLN migration object).

3.3.6.3 Migration Objects for Device Management

- **CONSUMPT (Change Period Consumption)**
The CONSUMPT migration object contains the fields to migrate the data for the period consumption. You can use it during to extrapolate the data that will be required in the future if no meter readings can be loaded for the corresponding prior periods.
A requirement for a migration of a period consumption is the successful migration of the corresponding billing related device installation (INST_MGMT migration object).
- **DEVICERATE (Change Rate Data for Installation Structure)**
The DEVICERATE migration object contains the fields for changing rates in an installation.
A requirement for a migration of a rate change is the successful migration of the corresponding installation (INSTLN migration object).
- **DEVICEREL (Create, Change, or Remove Device Allocations)**
The DEVICEREL migration object contains the fields defining, changing, or removing device allocations.
A requirement for a migration of a device allocation is the successful migration of the corresponding device installations (INST_MGMT migration object).
- **CONNCTINST (Installation Connection in Connection Object)**
The CONNCTINST migration object contains the fields for installing a connection in a connection object. This process entails the installation of a piece of equipment (connection) in a functional location (connection object).
A requirement for a migration of a device installation in a connection object is the successful migration of the corresponding connection object (CONNOBJ migration object) and connection (CONNECTION migration object).

- **INST_MGMT (Device Installation, Removal, or Replacement)**

The INST_MGMT migration object contains all the fields to install, remove, or replace a device in an installation or device location.

A requirement for a migration of a device installation is the successful migration of the corresponding device (DEVICE migration object), device location (DEVLOC migration object), and installation (INSTLN migration object).

- **METERREAD (Create Meter Reading)**

The METERREAD migration object contains the fields to create meter readings with the meter reading reasons 01 (periodic reading), 02 (interim reading with billing), 06 (move-in), 09 (interim reading without billing), 10 (control reading), 17 (change installation structure) or 19 (delivery reading).

A requirement for a migration of a meter reading is the successful migration of the installation of the corresponding device (INST_MGMT migration object).

Figure 3-7 displays the meter reading reasons which are available for a migration of meter readings. An existing meter reading order is required for some of the meter reading reasons. You cannot migrate a specific meter reading order. However, the application creates meter reading orders during some migration processes (for example, when migrating a move-out with the MOVE_OUT or MOVE_IN_H migration object).

Reading Reason	Meaning	Meter reading order required
01	Periodic reading	
02	Interim reading with billing	
03	Final meter reading at move-out	X
06	Meter reading at move-in	
08	Technical installation	X
09	Interim reading without billing	
10	Control reading	
12	Meter reading at technical removal	X
13	Meter reading on disconnection	X
17	Change of installation structure	
18	Meter reading on reconnection	X
19	Delivery reading	
21	Meter reading at billing-related installation	X
22	Meter reading at billing-related removal	X
23	Meter reading for device modification	X
24	Replacement upon removal	X

Figure 3-7: Meter Reading Reason which can be migrated

- **GOODSMVT (Goods Movement)**

The GOODSMVT migration object contains the fields for creating material documents for a goods movement. A material document with several items is posted during each call. You can migrate one or more serial numbers for each individual item. The first goods movement activity should be the creation of all the equipment and serial numbers currently in stock during the initial entry of stock balances (movement type 561).

A requirement for a migration of a goods movement is the successful migration of the corresponding device (DEVICE migration object).

- **OBJCLASS (Create Classification Data for Object)**
The OBJCLASS migration object contains the fields for creating the classification of technical objects. The Customizing of the migration object is set up for migrating classification data for the technical objects, such as, equipment or functional location.
A requirement for a migration of a classification is the successful migration of the corresponding technical object, such as, device (DEVICE migration object).
- **OBJSTATUS (Set User Status for Object)**
The OBJSTATUS migration object includes the fields for changing the user status of technical objects and profiles.
A requirement for a migration of an object status is the successful migration of the corresponding technical object, such as, device (DEVICE migration object).
- **REGRELSHIP (Create Register Relationships)**
The REGRELSHIP migration object contains the fields for creating register relationships.
A requirement for a migration of a register relationship is the successful migration of the installation of the corresponding device (INST_MGMT migration object).
- **STRT_ROUTE (Create Meter Reading Sequence)**
Use the STRT_ROUTE migration object to create street routes.
A requirement for a migration of a street route is the successful migration of the installation of the corresponding devices (INST_MGMT migration object).

3.3.6.4 Migration Objects for Disconnections

- **DISC_DOC (Create Disconnection Document)**
The DISC_DOC migration object contains the fields for creating disconnection documents.
A requirement for a migration of a disconnection document is the successful migration of the corresponding installation (INSTLN migration object) and, in dependency of the scenario to be migrated, device installation (INST_MGMT migration object), financial documents (DOCUMENT migration object), and dunning history (DUNNING migration object).
- **DISC_ORDER (Create Disconnection Order)**
The DISC_ORDER migration object contains the fields for creating disconnection orders. It is possible to create service orders or service notifications while creating disconnections orders.
A requirement for a migration of a disconnection order is the successful migration of the corresponding disconnection document (DISC_DOC migration object).
- **DISC_ENTER (Enter Disconnection)**
The DISC_ENTER migration object contains the fields for creating disconnection entries.
A requirement for a migration of a disconnection entry is the successful migration of the corresponding disconnection orders (DISC_ORDER migration object).
- **DISC_RCORD (Create Reconnection Order)**
The DISC_RCORD migration object contains the fields for creating reconnection orders.
A requirement for a migration of a reconnection order is the successful migration of the corresponding disconnection order (DISC_ORDER migration object).
- **DISC_RCENT (Enter Reconnection)**
The DISC_RCENT migration object includes the fields for creating reconnection entries.
A requirement for a migration of a reconnection entry is the successful migration of the corresponding reconnection order (DISC_ORDER migration object).

- **DISC_CLOSE (Complete Disconnection Document)**
The DISC_CLOSE migration object includes the fields for closing disconnection documents.
A requirement for a migration of a reconnection closure is the successful migration of the corresponding reconnection entry (DISC_RCEN migration object).
- **DISC_EVENT (Create DISCONNECT.CREATED Event)**
The DISC_EVENT migration object contains the fields for starting the disconnection workflow for a disconnection document by means of a workflow event. The created workflow event starts the disconnection workflow.
A requirement for a migration of an event for a disconnection workflow is the successful migration of the corresponding disconnection document (DISC_DOC migration object).
- **DISC_WF (Create Disconnection Order)**
The DISC_WF migration object contains the fields for starting the disconnection workflow for a disconnection document. The workflow is created directly without creating a workflow event.
A requirement for a migration of a disconnection workflow is the successful migration of the corresponding disconnection document (DISC_DOC migration object).

3.3.6.5 Migration Objects for Billing Documents

- **BILLDOC (Create Billing Document)**
The BILLDOC migration object contains the fields for creating billing documents based on a complete installation structure history. You can only reverse migrated billing documents using an adjustment reversal.
A requirement for a migration of a billing document is the successful migration of the corresponding contract (MOVE_IN or MOVE_IN_H migration objects) and device installation (INST_MGMT migration object).
- **BILLDOCEBF (Create Easy Bill Correction Framework (EBF) Billing Documents)**
The BILLDOCEBF migration object contains the fields for creating billing documents, which are not based on a complete installation structure history. You cannot reverse migrated EBF billing documents. You must use the EBF workbench to work on migrated EBF billing documents.
A requirement for a migration of an EBF billing document is the successful migration of the corresponding contract (MOVE_IN or MOVE_IN_H migration objects).
- **BILLTRIGCR (Create Billing Trigger)**
The BILLTRIGCR migration object contains the fields for generating billing orders for contracts with end of period billing or rolling back billing for monthly billings that have already taken place in the legacy system.
A requirement for a migration of a billing trigger is the successful migration of the corresponding contract (MOVE_IN or MOVE_IN_H migration objects).
- **CONSHIST (Create Billing Document for Consumption History)**
The CONSHIST migration object contains the fields for creating special, partial billing documents with consumption history only (DBERCHV structure).
A requirement for a migration of a consumption history in billing documents is the successful migration of the corresponding contract (MOVE_IN or MOVE_IN_H migration objects).

3.3.6.6 Migration Objects for EDM (Electronic Data Management)

- **LOADPROF (Create Load Profile for Installation)**

The LOADPROF migration object contains the fields for creating load profiles and allocating them to an installation.

A requirement for a migration of a load profile is the successful migration of the corresponding installation (INSTLN migration object).

- **PROFHEAD (Creating Header Data for EDM Profile)**

The PROFHEAD migration object contains the fields for creating EDM profile header data. You cannot create synthetic profiles and formula profiles using this migration object. EDM load profiles are migrated independently of other migration objects.

- **PROFASSIGN (Allocation of Register to EDM Profile)**

The PROFASSIGN migration object contains the fields to allocate profiles to registers of installed devices. Only active EDM profiles can be allocated. The unit of measurement and interval length must correspond in the profile header and the allocated register. You cannot change the allocation in periods that have already been billed.

A requirement for a migration of a load profile allocation is the successful migration of the corresponding installation (INSTLN migration object) and the device installation (INST_MGMT migration object).

3.3.7 Migration Objects Specific to SAP for Waste and Recycling

The following migration objects are specific to the industry solution SAP for Waste and Recycling as an extension of the industry solution SAP for Utilities:

- **CLEANOBJ (Create Cleaning Object)**

The CLEANOBJ migration object contains the fields for creating the cleaning object.

Cleaning objects are migrated independently of other migration objects.

- **AREA (Create Property)**

The AREA migration object contains the fields for creating the master data object property.

A requirement for a migration of a property is the successful migration of the corresponding cleaning object (CLEANOBJ migration object).

- **CONTAINER (Create Device or Equipment)**

The CONTAINER migration object contains the fields for creating the technical object equipment. The migration object contains the equipment data of the Plant Maintenance (PM) component as well as the data for the SAP for Waste and Recycling container. You can to create classification data of the container.

Containers are migrated independently of other migration objects.

- **CONTGRP (Create Container Group)**

The CONTGRP migration object contains the fields for creating container groups. It is possible to group a container already created in the system or to create a container during the creation of container groups with an automatic serialization.

A requirement for a migration of a container group can be the successful migration of the corresponding containers (CONTAINER migration object).

- **ROUTE (Create Route)**

The ROUTE migration object contains the fields for creating the master data object route.

Routes are migrated independently of other migration objects.

- **SERVREQ (Create Service Frequency)**
The SERVREQ migration object contains the fields for creating the master data object service frequency.
A requirement for a migration of service frequencies is the successful migration of the corresponding cleaning object (CLEANOBJ migration object), installation (INSTLN migration object), and route (ROUTE migration object).
- **PREMISEWA (Create Enhancements for Premise for SAP Waste and Recycling)**
The PREMISEWA migration object contains the fields that are specific to the master data object premise of the industry solution for SAP Waste and Recycling. It is possible to create multiple time slices.
A requirement for a migration of data for the enhancement of premise data is the successful migration of the corresponding premise (PREMISE migration object).

3.3.8 Migration Objects for Budget-Billing Plans

- **BBP_INT (Create Budget-Billing Plan from Scheduling)**
The BBP_INT migration object contains the fields for creating a budget-billing plan. Create a budget-billing plan according to the SAP for Utilities scheduling with a specified amount or with an amount projected by the system. The migration object can be used only for contracts that have been undergone a final billing in the legacy system. This is because you cannot model the real status of the legacy system (for example, there is no way to specify the due date or calculate an unequal distribution of the amounts due). You can only achieve an exact mapping of the budget-billing plan by using the BBP_MULT migration object. During the creation of a budget-billing plan with BBP_INT migration object, an internal billing simulation is carried out.
A requirement for a migration of a budget-billing plan is the successful migration of the corresponding contract (MOVE_IN migration object).
- **BBP_EXT (Create Budget-Billing Plan with External Scheduling)**
The BBP_EXT migration object contains the fields for creating a payment plan. You can only use this migration object to migrate payment plans (budget-billing procedure 03). In each case, you can only create one payment plan per contract. In case you need to create a budget-billing plan with statistics or debit position procedures (budget-billing procedure one or two), you must use the BBP_MULT migration object.
A requirement for a migration of a budget-billing plan is the successful migration of the corresponding contract (MOVE_IN migration object).
- **BBP_MULT (Create Budget-Billing Plan with External Scheduling)**
The BBP_MULT migration object contains the fields for creating a budget-billing plan with external scheduling. Unlike the BBP_EXT migration object, the BBP_MULT migration object creates budget-billing plans only with budget-billing procedure one or two (statistical or debit position procedure) for one or more contracts.
A requirement for a migration of a budget-billing plan is the successful migration of the corresponding contract (MOVE_IN migration object).
- **BBP_CB_TAB (Allocation of Collective Document Numbers)**
The BBP_CB_TAB migration object contains the fields for allocating a collective document number of budget-billing requests in the legacy system to a budget-billing plan in the SAP system. This allows the allocation of a payment for the collective budget-billing amounts, which have been requested in the legacy system but are paid in the SAP system. The customer pays the amount referring to the document number in the legacy system, which is stored in the FI-CA document as a reference document number.

A requirement for a migration of a collective document number is the successful migration of the corresponding open item (DOCUMENT migration object).

- **BBP_PAYSC (Budget-Billing Plan Payment Scheme)**

The BBP_PAYSC migration object contains the fields for creating a budget-billing payment plan.

A requirement for a migration of a budget-billing plan is the successful migration of the corresponding contract (MOVE_IN migration object).

3.3.9 Migration Object for Plant Maintenance (PM)

- **SM_NOTIF (Status Management (SM) Notifications Connection)**

The SM_NOTIF migration object contains the fields for a creation of service notification for a connection. After creation, the notification has the status *Created*.

A requirement for a migration of a service notification is the successful migration of the corresponding connections (CONNECTION migration object).

- **SM_NOTIFOK (Complete SM Notifications Connection)**

The SM_NOTIFOK migration object contains the fields for setting the status of a service notification to *Closed*.

A requirement for a migration of a service notification closure is the successful migration of the corresponding service notification (SM_NOTIF migration object).

- **TECHINST (Creating Technical Installation: Equipment)**

The TECHINST migration object contains the fields for creating the master data object technical installation. Creating a technical installation includes creating a piece of equipment and assigning it to a specified premise.

A requirement for a migration of a technical installation is the successful migration of the corresponding premise (PREMISE migration object).

- **TECHOBINSP (Creating Inspection Date for Technical Object)**

The TECHOBINSP migration object contains all fields for allocating the last inspection date to a device or a technical installation.

A requirement for a migration of an inspection date is the successful migration of the corresponding technical object, such as, device (DEVICE migration object) or technical installation (TECHINST migration object).

- **SALES_STAT (Sales Statistics)**

The SALES_STAT migration object contains the fields for creating sales statistics. It is necessary to create a respective information structure and generate the fields of the *ESTA* structure (use the fields of the information structure as a basis for this).

Sales statistics are migrated independently of other migration objects.

3.3.10 Migration Objects specific to SAP for Insurance

The following migration objects are specific to the industry solution FS-CD:

- **CD_INSOBJ (Create Insurance Object)**

The CD_INSOBJ migration object contains the fields for creating the master data object Insurance Object (IO).

A requirement for a migration of an insurance object is the successful migration of the corresponding business partners (PARTNER migration object). If you migrate an insurance object without an account creation variant, you must first migrate successfully the corresponding contract account (ACCOUNT migration object).

- **CD_VVSCPOS (Create Payment Plan Items)**

The CD_VVSCPOS migration object contains the fields for creating payment plan items.

A requirement for a migration of payment plan items is the successful migration of the corresponding insurance object (CD_INSOBJ migration object).

3.3.11 Migration Objects specific to SAP for Public Sector

The following migration objects are specific to the industry solution SAP for Public Sector:

- **PS_OBJ (Create Contract Object)**

The PS_OBJ migration object contains the fields for creating the master data object contract object.

A requirement for a migration of a contract object is the successful migration of the corresponding business partner (PARTNER migration object) and contract account (ACCOUNT migration object).

- **PS_OBJ_CHA (Change Contract Object)**

The PS_OBJ migration object contains the fields for changing the master data object contract object.

A requirement for a migration of a contract object change is the successful migration of the corresponding contract object (PS_OBJ_CHA migration object).

- **PS_RETURN (Tax return)**

The PS_RETURN migration object contains the fields for creating a tax return.

3.4 Generation of Own Migration Objects

SAP delivers a comprehensive set of migration objects to allow the data migration of most of the required data. All migration projects have special requirements that standard migration objects cannot always fulfill. Consequently, many migration projects need to create new migration objects. Although the IS Migration Workbench presents a complete set of functions to create new migration objects, some of them can be used only by specialists. To simplify the creation of new migration objects, the IS Migration Workbench offers the generation of new migration objects based on two different technologies:

- Batch Input (BDC Batch Data Communication)
- BAPI (Business Application Programming Interface)

For more information about how to generate new migration objects based on these technologies, see the following chapters.

3.4.1 Generation of a BDC Migration Object

A standard data transfer technology is batch input. A batch input session is a set of one or more calls to transactions along with the data that the transactions process. The system executes the transactions in a session non interactively, allowing rapid entry of bulk data into an SAP system. A session records transactions and data in a special format that the SAP system can interpret. When the system reads a session, it uses the data in the session to simulate online entry of transactions and data. The system can call transactions and enter data using most of the facilities that are available to interactive users. A batch input recording is a sequence of transactions with the screens they have run through. A session contains information about processed screens (program name, screen number), modified fields (field name and value), user commands executed (BDC_OKCODE), and the cursor position (BDC_CURSOR).

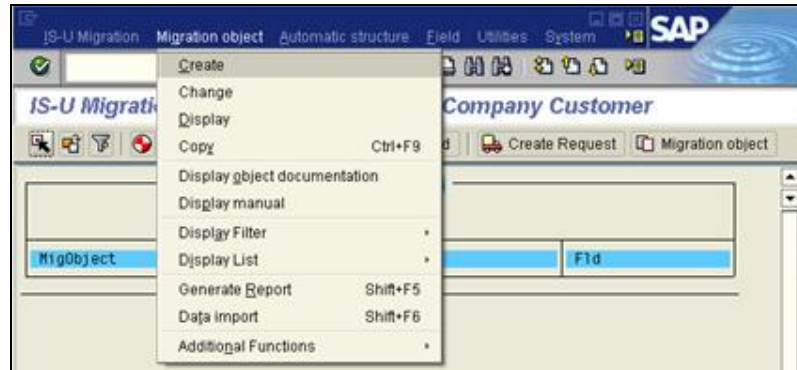
The IS Migration Workbench allows the generation of a migration object based on a recorded session. Instead of using the Data Transfer Workbench (transaction `BMV0`) to execute the data transfer, the IS Migration Workbench can be used for this purpose.

The generated migration object calls the `ISU_M_BDC_EXECUTE` service function module that passes the recorded information together with the actual field data to the ABAP statement `CALL TRANSACTION`. The advantage of using the IS Migration Workbench for the execution of a batch input recording is that the full set of functions of the workbench can be used, such as, field rules, error log, and distributed import.

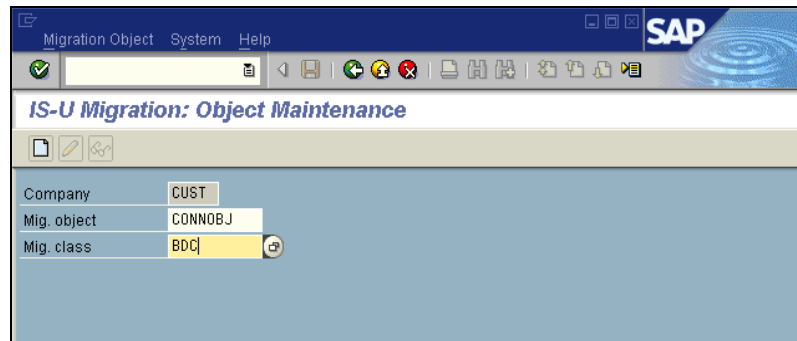
The Figure 3-8 shows the procedure for how to create a new migration object based on a batch input recording. After generating the migration object, you need to adjust the Customizing of the migration object. The IS Migration Workbench applies the *Transfer* field rule for all fields in the automation data.

A transaction returns the key of the newly created business object (when created with internal numbering) in a success message. The message is analyzed and the technical data of this message is entered as return field information. The content of the respective message variable is stored as the newkey in the KSM (Key and Status Management). Otherwise, the system uses the internal `DB_UPDATE` field (database update complete) to update KSM (for more information about the KSM, see chapter 2.5 Key and Status Management).

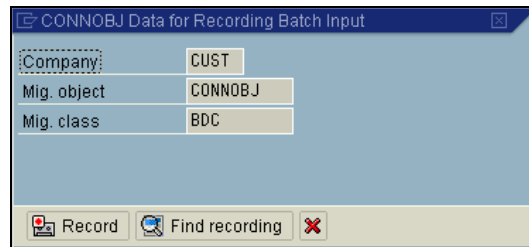
1. Change to own migration company and choose *Migration Object* → *Create*.



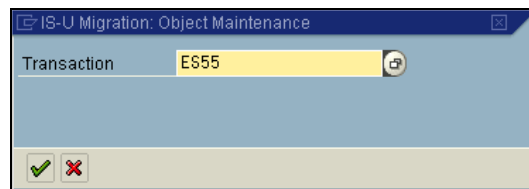
2. Enter the name of the new migration object and BDC as the migration class.



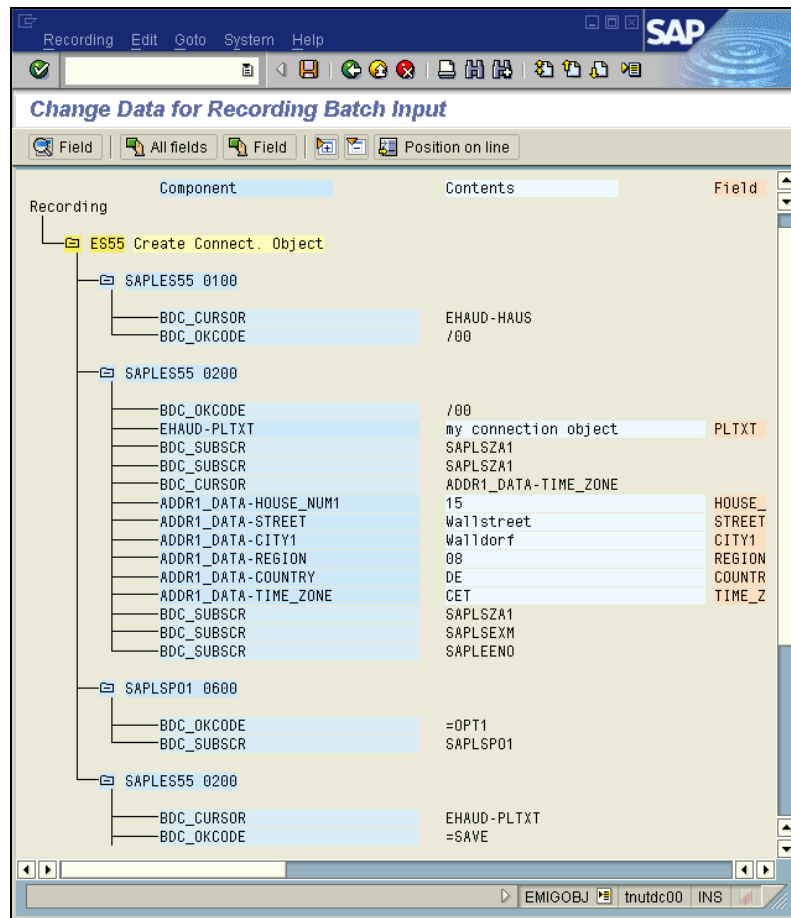
3. Choose the *Create* button.
4. Choose *Record* to create a new recording (or search for an existing one).



5. Enter the transaction code for the transaction to be recorded.



6. Choose *Enter* and execute the entered transaction by entering data in all fields required for data migration.
7. Choose *Save* when the transaction finishes.



8. Enter the abbreviation of the name of the migration object and the definition of the newly created migration object with the generated automation data structure based on the recorded transaction.

Company	CUST	Migration Company Customer
Mig. object	CONNOBJ	Mig. class BDC
Mig. obj. text	Migration Object CUST - CONNOBJ	
Mig. obj. abbr.	COB	3 Create
Blocking Status	000	Do not block any

Object Data	Relationship data	Parameters
Service module ISU_M_BDC_EXECUTE Interface		
Auto struc.type ZMG_CUST_CONNOBJ_AUTO		
Transaction ES55 New Recording Recording data		

Return field	DB_UPDATE
Or	
ApplicationArea	E9
Message Number	004
Mess. variable	1

9. The newly migrated object with its automation structure is now displayed

WB status		
Object:	CONNOBJ	Migration Object CUST - CONNOBJ
Struct.:	DATA	Automation Data for BDC Object CUST - CONNOBJ
MigObject	Auto.Struct.	FId
CONNOBJ	DATA	
		CITY1_001 3
		COUNTRY_00 3
		HOUSE_NUM1 3
		PLTXT_001 3
		POST_CODE1 3
		REGIOGROUP 3
		REGION_001 3
		STREET_001 3
		TIME_ZONE_ 3

Figure 3-8: Procedure to Create a New BDC Migration Object

Batch input is fully supported with release 4.6. However, there are new transactions that are no longer supported if ActiveX controls are used. In most cases, the SAP still provides and supports the old transactions alongside the new ones. This ensures that you can also use customer-defined and SAP standard batch inputs as of release 4.6. But, this is not a long-term strategy for SAP, which is why we are continuing to push the development of new BAPIs that will gradually replace the current batch inputs. For more information about batch input, see the documentation of the transaction SHDB.

3.4.2 Generation of a BAPI Migration Object

Starting with the release ERP2005, the IS Migration Workbench allows the generation of migration objects using existing function modules as service function modules. Initially, it was designed to allow a data migration with the IS Migration Workbench also using BAPIs. However, this function allows the generation of migration objects based on any function module, which follows the design rules of a BAPI. These design rules are as follows:

- The interface of the function module may contain either fields or flat structures as import, export, or changing parameters. It is not possible to define table types as importing or exporting parameters.
- The interface of the function module may contain table parameters based on flat structures.
- The function module may execute neither a COMMIT WORK nor a ROLLBACK WORK.
- The interface must include a parameter named RETURN, either as an export parameter or as a table parameter. The return parameter may be based on one of the reference structures BAPIRET1, BAPIRET2, or BAPIRETURN. The return parameter must be filled in the logon language (for example, in using the BALW_BAPIRETURN_GET2 function module) and report any error that might have occurred during processing to the calling program.
- The interface must not include exceptions.

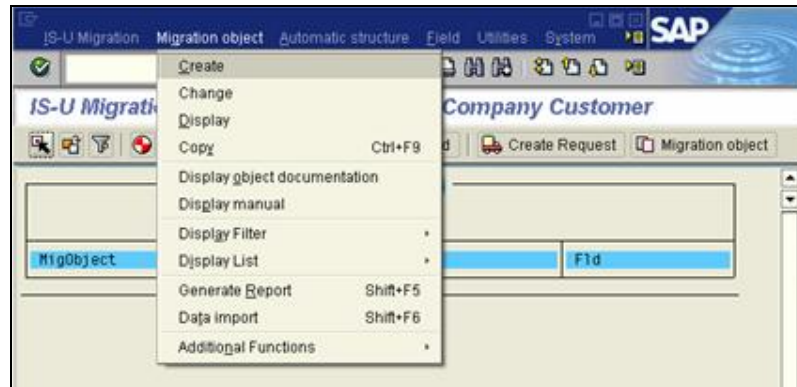
During the generation of the BAPI migration object, the IS Migration Workbench analyses the parameter interface of the service function module. If the RETURN parameter can be found in the parameter interface, the IS Migration Workbench adds an event at the CALL01 generation point to generate in the load report a call of the ISU_M_BAPI_MESSAGE_PROCESSING function module. This function module analyses the error messages in the RETURN parameter as returned by the service function module of the BAPI migration object. The ISU_M_BAPI_MESSAGE_PROCESSING function sets the DB_UPDATE parameter to X if no error message is returned, otherwise to space. For more information about the usage of the DB_UPDATE parameter, see chapter 7.2.6 *Error EM 104*:

Table x not supplemented. For more information about generation points and events, see chapter 4.3.3 *Code on Report Level*.

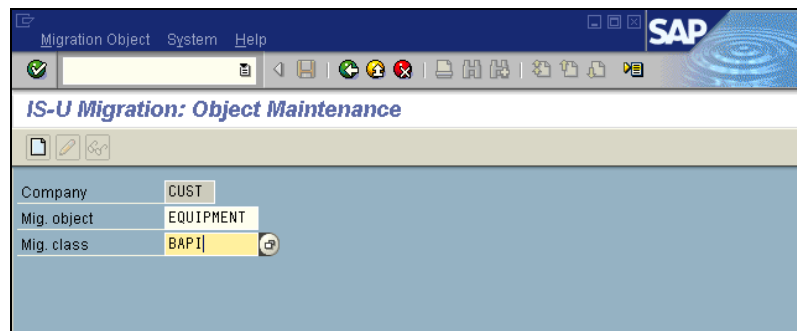
You must recreate a BAPI migration object after you have changed the parameter interface of the service function module, for example, in adding or deleting a parameter.

Figure 3-9 shows the procedure to create a new migration object based on a function module.

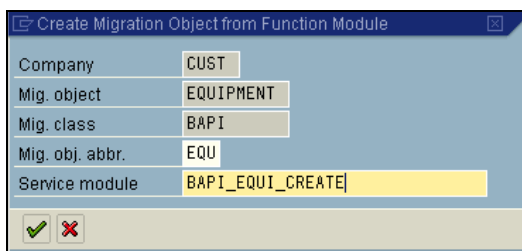
1. Change to your own migration company and choose *Migration Object* → *Create*.



2. Enter the name of the new migration object and BAPI as the migration class.



3. Push the *Create* button
4. Enter the abbreviation of the name of the migration object and the name of a BAPI.



5. Choose *Enter*

Company: CUST Migration Company Customer
 Mig. object: EQUIPMENT Mig. class: BAPI
 Mig. obj. text: PM BAPI: Create Equipment
 Mig. obj. abbr.: EQU 3 Create
 Blocking Status: 100 Data import blocked

Object Data Relationship data Parameters Software Component

Service module: BAPI_EQUI_CREATE Interface
 Auto struc.type: ZMG_CUST_EQUIPMENT_AUTO Mult. leg. keys
 Return structure: BAPI_NEW_KEY Structure type: TEMKSV
☐ Internal Table
☐ Structure type
 Return field: BAPI_NEW_KEY-NEWKEY

6. Save the definition of the newly create migration object with the generated automation data structure based on the interface of the BAPI and return to the main screen.
7. The newly migrated object with its automation data structure is now displayed.

MigObject	Auto Struct.	Fld
EQUIPMENT	IMPORT_FIELDS	EXTERNAL_N 3
	DATA_GENERAL	VALID_DATE 3
	DATA_SPECIFIC	
	DATA_FLEET	
	DATA_INSTALL	
	EXPORT_FIELDS	
	DATA_GENERAL_EXP	
	DATA_SPECIFIC_EXP	
	DATA_FLEET_EXP	
	RETURN	

Figure 3-9: Procedure to Create a New BAPI Migration Object

You need to adjust the Customizing of the migration object after the generation of the migration object. The IS Migration Workbench applies the *Transfer* field rule for the fields in the automation data. If the service function module returns a key that should be used as a reference to other migration objects, you have to change the **Return field** field on the object maintenance screen. The BAPI BAPI_EQUI_CREATE returns the equipment number of the created equipment in the exporting EQUIPMENT interface parameter. You can change the return field from BAPI_NEW_KEY-NEWKEY to AUTO-EXPORT_FIELDS-EQUIPMENT to store the equipment number in the KSM.

Return field: AUTO-EXPORT_FIELDS-EQUIPMENT

Figure 3-10: Changed Return Field



Enter DB_UPDATE in the **Return field** field if the service function module does not return a key that can or should be used as a reference to further migration objects.

3.5 Hyper Object

3.5.1 Motivation

A hyper object is a migration object with the HYPER migration class. A hyper object can be used to import data of several migration objects within one import file. In this way, you can import dependent data that is described in separate migration objects at the same time. In a hyper object, you create references to existing migration objects. The settings for the referenced migration objects are transferred to the loading report for the hyper object when the hyper object is generated.

Hyper objects are implemented when you need to migrate an installation structure history (industry solution SAP for Utilities). For this, a sequence of transaction data has to be migrated in chronological order (for example, device installation, device removal, device relationship, and meter readings).

3.5.2 Restrictions

Using hyper objects leads to limitation. Due to the nature of hyper objects, a parallel data load is not possible. The dependency between a hyper object and the migration objects referenced in the hyper object, causes the following restrictions when creating, maintaining, and migrating data with a hyper object and its referenced migration objects:

- The maintenance of auto structures and fields of a referenced migration object is only possible in the referenced migration object. Changes made in the referenced migration object are automatically transferred into the hyper object after saving. The settings for the auto structures and fields can only be displayed in the hyper object. A comparison with the ABAP dictionary (DD) and new positioning of fields is automatically executed in the referenced migration object.
- You cannot copy hyper objects.
- The transport of a hyper object includes all referenced migration objects, but the transport of a referenced migration object includes the transport of the relevant hyper objects.
- You cannot create references for migration objects from the HYPER and NOTE migration classes.
- You cannot upload migration Customizing for a hyper object or for a migration object while it is referenced in a hyper object.
- The following procedures can no longer be executed:
 - Change of the short description of the migration object
 - Change of the auto structure type
 - Creation and deletion of auto structures
 - Maintenance of parameters for an auto structure (exceptions are: **Generation** field and **Indicator** field)
 - Deletion of the migration object; this is only possible once you have deleted all hyper objects that have a reference to the migration object you want to delete.
- It is not possible to display migrated data objects from the file editor using the **Display Object** button.
- When you migrate objects with a hyper object, the KSM is only updated for the referenced migration objects. Therefore, no KSM entries are available for a hyper object.

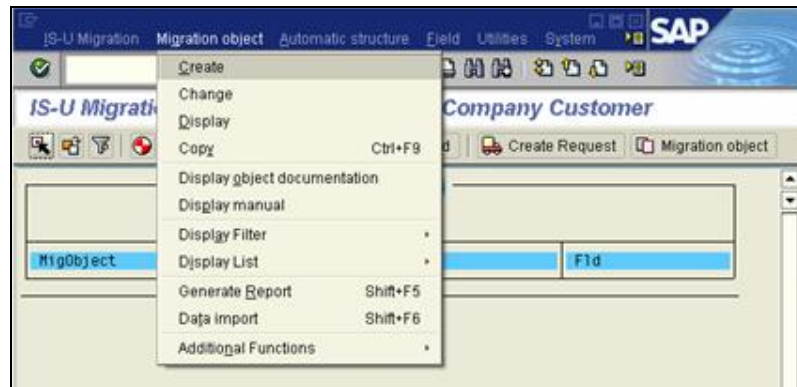


We recommend that you activate the locks for dependent object data when you are importing data for hyper objects. Locks prevent the importation of data that you cannot consistently import for a recall or due to incorrect data in the previously processed data records. For more information, see chapter 2.8 *Migration Lock Mechanism*.

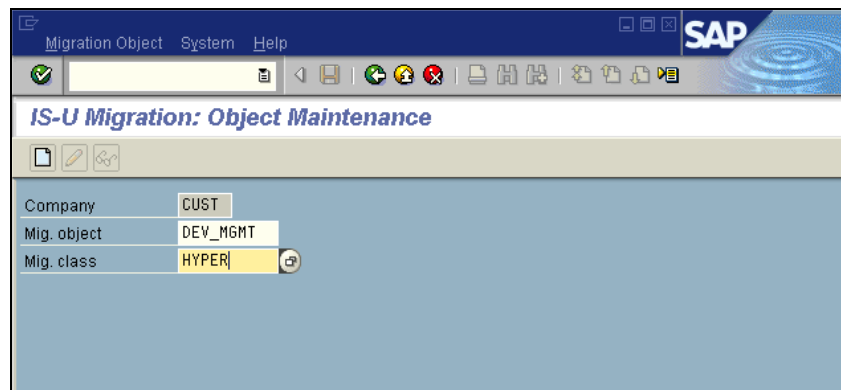
3.5.3 Creation Procedure (Hyper Object)

Figure 3-11 shows the procedure for creating a new migration object as a hyper object.

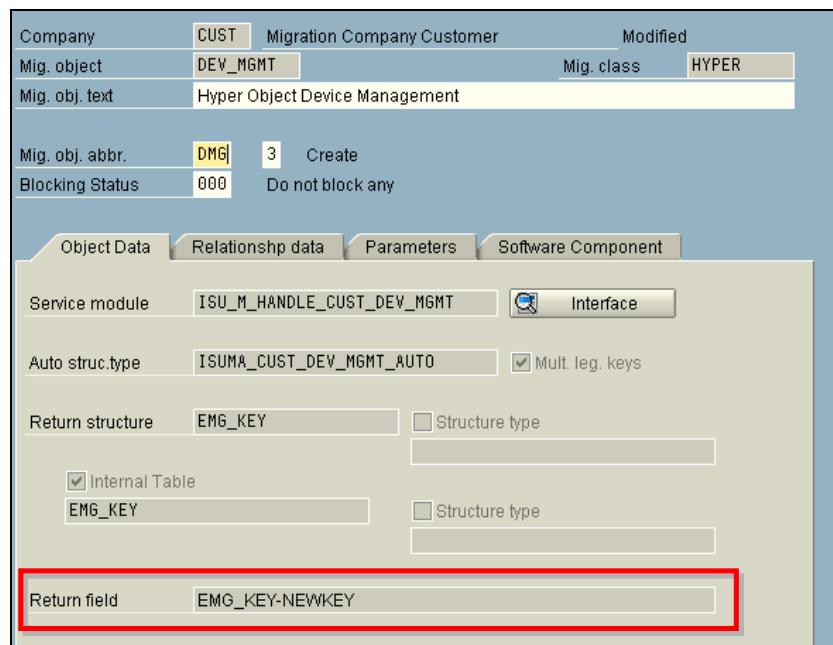
1. Change to own migration company and choose *Migration Object* → *Create*.



2. Enter the name of the new migration object and enter HYPER as the migration class.



3. Push the *Create* button.
4. Enter the name of a BAPI and the abbreviation of the migration object and save the definition of the hyper object.



5. Save the definition of the newly created migration object with the generated automation data structure, based on the interface of the BAPI.

6. The newly migrated hyper object can now be displayed.

Object: DEV_MGMT Hyper Object Device Management		
MigObject	Auto.Struct.	FId
CONNBJ		
DEVGRP		
DEVICE		
DEVICEREL		
DEVLOC		
DEV_MGMT		
INSTLN		
INST_MGMT		
METERREAD		
PREMISE		

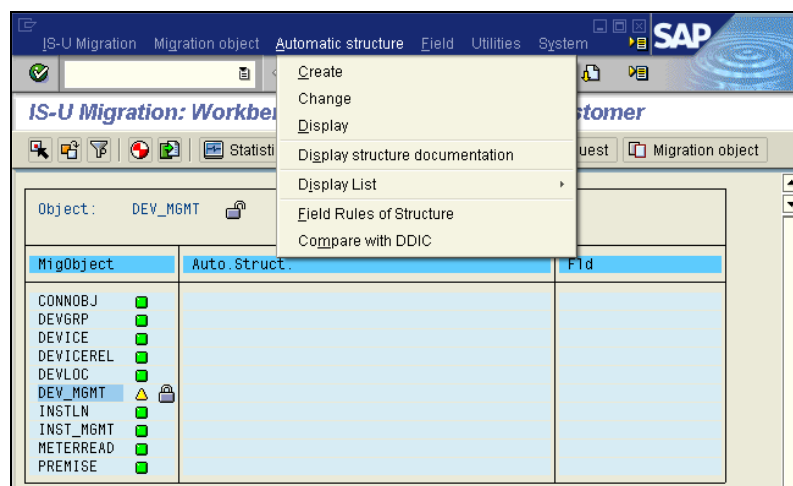
Figure 3-11: Procedure to Create a New BAPI Migration Object

3.5.4 Creation Procedure (Reference to Migration Object)

The hyper object is created without an automation structure. You define the automation structure when you create references to existing migration objects. Figure 3-12 shows the procedure for adding migration objects as a reference in the created hyper object. Arrange the auto data of the referenced migration object below the created automatic structure with the name of the migration object.

When you create an automation structure from the **MigObject** automatic structure type, new structures for the hyper object are automatically generated for each referenced migration object. The new structures include the automation structure of the referenced migration object. Subsequently, the system checks the object specific ABAP code (for example, the code in the field rules and events) when creating the automation structure in the hyper object. If you must reference several migration objects in one hyper object, check that the same data variable definitions do not exist more than once in the field rules. This causes syntax errors when you generate the load report for the hyper object. Use the prefix **LV_** (local variable) for the definition of local data objects. This prefix is replaced in the hyper object by the combination of **LV_** and the short name of the referenced migration object. For more information about the generation of the load report and the replacement rules, see chapter 3.5.5 *Generation of the Load Report*:

1. First, select the hyper object and choose *Automation Structure → Create*.



- Choose the *MigObject* automatic structure type and enter the name of the migration object to be referenced. A hyper object cannot be referenced by another hyper object.

- Choose *Create*.

- Choose *Save* and return to the main screen to display the added referenced migration object.

WB status		
Object: DEV_MGMT Hyper Object Device Management		
MigObject	Auto. Struct.	Fld
CONNBJ	INST_MGMT	
DEVGRP	INTERFACE	
DEVICE	AUTO_ZW	
DEVICEREL	AUTO_GER	
DEVLOC	CONTAINER	
DEV_MGMT		
INSTLN		
INST_MGMT		
METERREAD		
PREMISE		

- To add more referenced migration objects, repeat the steps one to four for all required migration objects (DEVICEREL, METERREAD and DEVGRP).

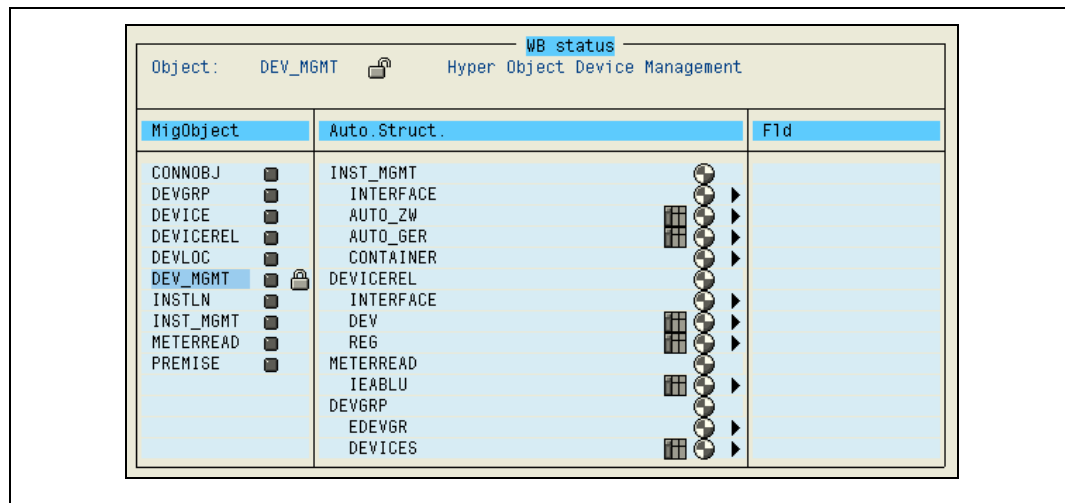


Figure 3-12: Procedure to Create a Reference for a Migration Object

3.5.5 Generation of the Load Report

There is a HYPER migration object available in the migration company SAP. This migration object and its events are used to create a new hyper object. The load program generator of the IS Migration Workbench generates a load program as follows:

- Generation of the type of the auto structure **AUTO**
The load program generator uses the automation data of the referenced migration objects to generate the auto structure type of the hyper object. The components of this type are generated based on the auto structures of the referenced migration objects.
- Generation of the type of the auto structure **INFO**
The load program generator checks all auto structures of both the hyper object and the reference migration object whether they have the customer structure indicator marked. Such structures are generated in the customer structure type.
- Generation of the code from the field rules
The load program generator analyzes the ABAP code in the field rules and makes replacements according to the replacement rules described in Figure 3-13 when the code is transferred to the hyper object. The replacements already happen either when you create a reference of a migration object or when you make changes to the code stored in the field rules of the referenced migration object.
- Generation of the service function module
The load program generator generates a service function module for the hyper object in which the calls of the service function modules of the referenced migration objects are included. It inserts the ABAP code of the events of the generation points (CALL01 or CALL02) of each of the referenced migration objects before and after the generated call of the respective service function module. The load program generator analyzes the specific structure names of the transfer parameters of the service function modules and the code of the inserted events for each referenced migration object and makes replacements according to the replacement rules described in Figure 3-13.
- Generation of the load report
The load report is generated from the settings of the hyper object and the created auto structures. In the load report, the load program generator inserts the code of events for the hyper object and the referenced migration object as follows:

- At the **INCLUDE generation point** (Integration of include statements at report start), the include programs of the hyper object are transferred to the load report, and then to those of the referenced migration object with an `INCLUDE` statement. `INCLUDE` programs with identical names are taken into account only once.
- At the **CSTINCL generation point** (Integration of additional include statements at report start), the include programs of the hyper object are transferred to the load report, and then to those of the referenced migration object with an `INCLUDE` statement. Include programs with identical names are taken into account only once.
- At the **FILL01 generation point** (before processing of field rules for a data record), the events of the hyper object and the events of the respective referenced migration objects are inserted into the load report. Events of the referenced migration object are only taken into account if they are defined with a category other than *Standard*.
- At the **FILL02 generation point** (after the field rules for a data record are processed), the events of each referenced migration object are transferred to the load report, and then to those of the hyper object. Events of the references migration objects are only taken into account if they are defined with a category other than *Standard*.
- At the **SERVIO1 generation point** (before the service function module of the hyper object is called up), the events of the referenced migration objects within a case distributor are inserted into the load report through a query of the current referenced migration object. The events of the hyper object are inserted. Take the events of the referenced migration objects into account if they are defined with a categorie other than *Standard* or *BDC*.
- At the **SERVIO2 generation point** (after the service function module of the hyper object is called), the events of the hyper object are inserted in the load report. Insert the events of the referenced migration objects within a case distributor through a query of the current referenced migration object. Take the events of the reference migration objects into account if they are not defined with the categories *Standard* and *BDC*.

For more information about generation points and their placement in the load report, see chapter 4.3.3 *Coding Report Level*.

○ Replacement of structure and field names

Before the code from field rules and events of the reference migration objects is transferred to the hyper object, the code is analyzed and adjusted to comply with the environment of the hyper object. Figure 3-13 shows the replacement rules.

Code in Referenced Migration Object	Changed Code in Hyper Object
Structure name AUTO	AUTO-<Name of the referenced migration object>
Structure name of a customer structure	INFO-<Name of the customer structure>
Prefix of local variable LV_	LV_<ID of referenced MigObject>_
Prefix of local internal table LT_	LT_<ID of referenced MigObject>_
In field rules: \$C\$ = Z_<Name of input structure>	\$C\$ = Z_<Name of input structure in hyper object>

Figure 3-13: Rules for Replacement of Code in Hyper Object

3.6 Migration Control Parameter

Sometimes, it is preferable to change the way the application processes the migration data. The most important reason is to improve the performance of the data load by avoiding checks in the application that are not necessary, for example, due to the data migration strategy. It might also be that the SAP application must accept the data because the data was already accepted in the legacy system and needs to be transferred as it is.



Verifying the existence of billing documents while you are migrating device installations may not be required if, according to the migration strategy, billing documents are to be migrated after the installation of devices. In bypassing this verification, you can save database accesses. This improves the performance of the device installations. However, if billing documents already exist due to a previous migration step, a verification must be executed.



The requirements are different in each project. Therefore, it is in the responsibility of the project to verify whether it is possible to take advantage of a control parameter. Switching off standard checks may lead to inconsistencies in the application and the database. Use control parameters carefully. Documentation is available for each control parameter. Check the documentation in the IS Migration Workbench thoroughly before changing the delivered settings.

The application reads the control parameter during data migration, and treats the data according to the setting of the relevant control parameter. Sometimes, the setting of the control parameters supersedes the Customizing in the IMG. The following control parameters are available in the IS Migration Workbench:

3.6.1 General Control Parameters

- **FICA_INTEREST_DATE** (Definition of a field in the `FKKOP_T` structure)
The `FICA_INTEREST_DATE` control parameter decides which field of the `FKKOP` structure the information of the last interest calculation date in the legacy system is transferred to. This occurs during data migration of financial open items. When open items with the `DOCUMENT` migration object are migrated, a corresponding interest calculation attachment is written to ensure correct processing of the document in the next interest calculation run.
- **EVENT_DISTR_IMPORT** (User-specific function module for distributed import)
The `EVENT_DISTR_IMPORT` control parameter defines a custom-specific function module during the execution of the distributed import. For more information, see chapter 2.7.5 *User-Exit in Master Job*.

3.6.2 Control Parameters Specific to SAP for Utilities

- **BBP_DFKKTHI_OFF** (Deactivation of the update of the `DFKKTHI` database table)
The `BBP_DFKKTHI_OFF` control parameter controls the update of `DFKKTHI` database table during the migration of budget-billing plans. The application updates the `DFKKTHI` database table when deregulated budget-billing plans are migrated even a dispatch or aggregated posting is not required. For more information, see SAP note 794199.
- **INST_CERT_CHECK_OFF** (Deactivation of a certification check for a device installation)
The `INST_CERT_CHECK_OFF` control parameter controls the certification of devices during the migration of device installation. This parameter prevents checks for the next replacement year during device installation so it is possible to migrate historic processes correctly. For more information, see SAP note 314579.

- **INST_CERT_LOSS_OFF** (Avoidance of the loss of the device certification)
The INST_CERT_LOSS_OFF control parameter controls the certification of devices during the migration of device removal. This parameter is designed to avoid the loss of certification during the removal of devices (without considering the settings in Customizing), so it is possible to display historic processes correctly. For more information, see SAP note 314579.
- **INST_BILLPER_OFF** (Deactivate check for billed periods)
The INST_BILLPER_OFF control parameter controls the checks of already billed periods during the migration of device installation, removal, and replacement. This parameter allows changes of the installation structure in periods where billing documents already exist for this installation. For more information, see SAP note 449827.
- **INST_CONS_OBJ_OFF** (Deactivate period consumption for device installation)
The INST_CONS_OBJ_OFF control parameter controls the processing of the period consumption during the migration of device installation, removal, and replacement. This parameter deactivates the processing of the period consumption object. The period consumption must be migrated separately with the CONSUMPT migration object. For more information, see SAP note 314579.
- **INST_CONS_BILL_OFF** (Deactivate period consumption during bill-related installation)
The INST_CONS_OBJ_OFF control parameter controls the processing of the period consumption during the migration of a billing related device installation. This parameter deactivates the processing of the period consumption object. For more information, see SAP note 445399.
- **INST_CONS_RATE_OFF** (Deactivate period consumption during rate maintenance)
The INST_CONS_RATE_OFF control parameter controls the processing of the period consumption during the migration of rate data at device and register level with the DEVICERATE migration object. This parameter suppresses a generation of time slices when migrating rate data if the period consumption values have already been migrated.
- **INST_DISMANT_CH_OFF** (Deactivate check for disconnection)
The INST_DISMANT_CH_OFF control parameter controls the checks for disconnections during the migration of a device installation, removal, and replacement. This parameter deactivates the checks for device disconnection to be able to work with devices regardless of the assigned disconnection status. For more information, see SAP note 445399.
- **INST_PREDEF_EAST_OFF** (ignore predefined register relationship)
The INST_PREDEF_EAST_OFF control parameter controls the processing of predefined register relationships during the migration of device installation. This parameter deactivates the consideration of predefined register relationship in Customizing. The register relationship can be migrated separately with the REGRELSHIP migration object. For more information, see SAP note 445399.
- **INST_FACTS_REQ_OFF** (Deactivate Check for Required Value for Operand)
The INST_FACTS_REQ_OFF control parameter controls the handling of the *Required Value for Operand* indicator during the migration of business objects that effect installations. With this control parameter, missing required values for operands are ignored when you create or change installations until the migration of operands with the FACTS migration object.
- **INST_POD_OBJ_OFF** (Deactivate changes to PoD during changes to installation)
The INST_POD_OBJ_OFF control parameter controls the processing of a point of delivery (PoD) during the migration of changes of an installation. This parameter deactivates the processing of the POD when an installation is changed. For more information, see SAP note 508222.

- **INST_POD_PRO_CH_OFF** (Deactivate the profile and PoD allocation)
The INST_POD_PRO_CH_OFF control parameter controls the checks of profile and PoD allocations during the migration of device installation. This parameter deactivates the checks for existing profile and POD allocations. For more information, see SAP note 445399.
- **MOI_POD_SERV_CH_OFF** (Deactivate default values of PoD services during move-in)
The MOI_POD_SERV_CH_OFF control parameter controls the processing of non-billable PoD services during the migration of a move-in with the MOVE_IN migration object. By not carrying out a check for existing default values and subsequent allocation of non-billable POD services, the customized proposal logic of POD services is deactivated. In addition, transferred POD data in the automation data is not processed. POD services have to be migrated separately with PODSERVICE migration object. The usage of the MOI_POD_SERV_CH_OFF control parameter improves the performance of the migration of move-ins. For more information, see SAP note 508222.
- **MOI_MR_ORDER_OFF** (Deactivate creation of meter reading order during move-in)
The MOI_MR_ORDER_OFF control parameter controls the creation of a meter reading order during the migration of a move-in with the MOVE_IN migration object. Meter readings with meter reading reason 06 (move-in) have to be migrated either separately or implicitly when installing a device billing related. The usage of the MOI_MR_ORDER_OFF control parameter reduces the performance of the migration of move-ins. For more information, see SAP note 1367103.
- **MR_VALIDATION_OFF** (Deactivate meter reading result validation)
The MR_VALIDATION_OFF control parameter controls the processing of meter reading validation during a device installation with INST_MGMT migration object, and the migration of meter readings with METERREAD migration object. This parameter allows the migration of meter readings, which have been accepted in the legacy system but would be rejected by the defined plausibility rules in the Customizing of the SAP system.
- **POD_BILLPER_OFF** (Deactivate checks of already billed periods)
The POD_BILLPER_OFF control parameter controls the checks of already billed periods during the migration of meter readings. This parameter allows changes to the PoD for periods that have already been billed. For more information, see SAP note 656893.
- **READ_BILLPER_OFF** (Deactivate checks of already billed periods (meter reading))
The READ_BILLPER_OFF control parameter controls the checks of already billed periods during the migration of meter readings. This parameter allows the migration of meter readings in periods that have already been billed, for example, the migration of meter readings after the successful migration of billing documents. For more information, see SAP note 449827.
- **PERF_EVENT_OFF** (Deactivate update of workflow events of SAP for Utilities)
The PERF_EVENT_OFF control parameter controls the handling of workflow events during the data migration for all SAP for Utilities specific migration objects.
- **PERF_STATS_OFF** (Deactivate update of stock/transaction statistics)
The PERF_STATS_OFF control parameter controls the creation of stock and transaction statistics during the data migration for all relevant SAP for Utilities specific migration objects (material management or plant management).
- **CRM_REPL_OFF** (Deactivation of CRM replication)
The CRM_REPL_OFF control parameter controls the delta replication of business objects to the SAP CRM system during data migration. For more information, see chapter 6.18.2.2 *Replication Strategy: Replication during Data Migration*.

3.6.3 Configuring Control Parameters

Choose *IS Migration* → *Settings* → *Customer Settings* for a configuration of the control parameters. Define control parameters for all configured migration companies by leaving the **Company** field empty. If you enter a migration company but leave the **Mig.Object** field empty, the setting of the control parameter is valid for the entered migration company only. By entering a migration company and a migration object, you make the setting of the control parameter migration object specific. Figure 3-14 shows the maintenance screen for control parameters.

Figure 3-14: Control Parameter for Custom-Specific Function Module

Figure 3-15 shows defined control parameters in a customer project.

Com...	Mig. object	Ctrl parameters	Ctrl Parameter Value	Ctrl Parameter Text
		EVENT_DISTR_IM...	ISU_M_SAMPLE_DISTR_IMPO...	
CUST		CRM_REPL_OFF	X	Deactivate CRM Replication (only IS-U)
CUST	MOVE_IN	MOI_POD_SERV_C...	X	Deactivate Default Values of POD Services Du

Figure 3-15: Customer Settings of Control Parameters

3.6.4 Function Module ISU_MIG_STATUS

The application calls the function module `ISU_MIG_STATUS` to check the settings of the migration control parameters. The interface of the function module `ISU_MIG_STATUS` has the following exporting parameters:

- **Y_ACTIVE** Application is called by a load program (`Y_ACTIVE = 'X'`)
- **Y_FIRMA** Name of the migration company
- **Y_OBJECT** Name of the migration object whose load program is executed
- **Y_TEMSTAT** Structure with all control parameters and their actual settings
- **Y_<parameter>** Specific control parameter

You can use this function module to bypass implemented user-exits and BADIs (business add-in) which do not need to be executed during data migration (for example, to improve the migration performance). Figure 3-16 shows a code snippet that calls the function module `ISU_MIG_STATUS` in a customer code. The customer code is executed when migrating business partners with the load program of the `PARTNER` migration object.

```
* local data definition
DATA: lv_active TYPE c, lv_object TYPE emg_object.

* check whether data migration is active
CALL FUNCTION 'ISU_MIG_STATUS'
  IMPORTING
    y_active = lv_active
    y_object = lv_object.
IF lv_active = 'X' AND lv_object = 'PARTNER'.
* execute migration specific code
ELSE.
* execute non-migration specific code
ENDIF.
```

Figure 3-16: Usage of the function module ISU_MIG_STATUS

4 Enhancing a Load Program

4.1 Motivation

This chapter shows how to handle some standard tasks when using the IS Migration Workbench for enhancing the load report or generating own migration objects. It describes the layout of the generated load program and details where and how own ABAP code can be inserted. Finally, the chapter describes how to incorporate customer fields in the data migration process.

4.2 Layout of the Load Program

SAP does not deliver load programs. In addition, in the SAP system, no interpretation of the field rules takes place during data migration. Instead, a load program is generated by the IS Migration Workbench (*Migration Object* → *Generate Load Program*). As described in chapter 2.1 *Load Program*, the load program generator converts the Customizing of a migration object (for example, field rules and events) into an executable report. An event in a load program is defined by its type, the generation point in the load program, and the sort order within all events of a generation point (for more information, see chapter 4.3.3 *Code on Report*). Choose *Migration Object* → *Change* to display the migration object maintenance screen. Choose *Utilities* → *Events* to display and change events of the migration object. Figure 4-1 shows a migration object with its defined generation points and events.

Gen. Point	Sort	Icon	Coding Tp	Description	Category	Variant	Code
INCLUDE	10	Include		RE3DAALL	Standard		
	40	Include		RE3FOALL	Standard		
CSTINCL							
START	10	Perform		GET_SYSTEM_CPAGE	Standard		RC_EXIT
	20	Perform		MSG_OPEN	Standard	1	
	30	Perform		OPEN_FILE	Standard	1	RC_EXIT
	40	Perform		OPEN_FILE	Standard	2	RC_EXIT
	50	Perform		BATCH_MSG	Standard		
	60	Perform		INIT_ITYP	Standard		
	70	Perform		CREATE_STAT	Standard		
	75	Perform		CHECK_AUTH	Standard		RC_EXIT
	80	Perform		CHECK_COMMIT	Standard		
	90	Perform		DEAKTIVATE_PARAMETER	Standard		
	100	Perform		HOLD_TIME	Standard	1	
	110	Funktion		ISU_M_KSV_PUFFER	Buffering	1	SYRC_EXIT
	120	Funktion		ISU_M_T_RUNTIME_START	Performance		
READ01	10	Perform		READ_DATA	Standard		RC_EXIT
FILL01	10	Perform		CHECK_ITYP	Standard	1	RC_EXIT
FILL02							
OTHERS	10	Perform		CASE_OTHERS	Standard		K_EXIT
	20	Perform		CASE_OTHERS_EVENT	Standard	1	K_EXIT
READ02							
SERVI01	10	Perform		CHECK_ERROR	Standard		K_CONTINUE
	20	Perform		TEMP_MSG_OPEN	Standard		
CALL01							
CALL02							
SERVI02	10	Perform		SERVICE_ERROR	Standard	1	K_CONTINUE
	10	Perform		COMMIT_WORK	Standard		
SERVI03	20	Funktion		ISU_M_T_RUNTIME_STOP	Performance		
	30	Funktion		ISU_M_T_RUNTIME_START	Performance		
LAST	10	Funktion		ISU_M_KSV_INSERT_LAST	Buffering	1	
	10	Perform		CLOSE_STAT	Standard		
END	20	Perform		CLOSE_FILE	Standard	1	
	30	Perform		CLOSE_FILE	Standard	2	
	40	Perform		HOLD_TIME	Standard	2	
	50	Perform		MSG_CLOSE	Standard	1	
	60	Funktion		ISU_M_T_RUNTIME_EXPORT	Performance		

Figure 4-1: Generation Points and Events of a Migration Object

The most important generation points of a load program are:

- **CSTINCL** Includes additional include statements at the start of the report
- **START** Before the data records are imported and processed
- **FILL01** Before processing the field rule for a data record
- **FILL02** After processing the field rule for a data record
- **SERVI01** Before the service function module is called
- **CALL01** Directly before the service function module is called
- **CALL02** Directly after the service function module is called
- **SERVI02** After the service function module has been called
- **SERVI03** After a data object has been created
- **LAST** End processing of a data object
- **END** Actions after the last data record has been processed

Figure 4-2 shows the layout of the load program in its generation points.

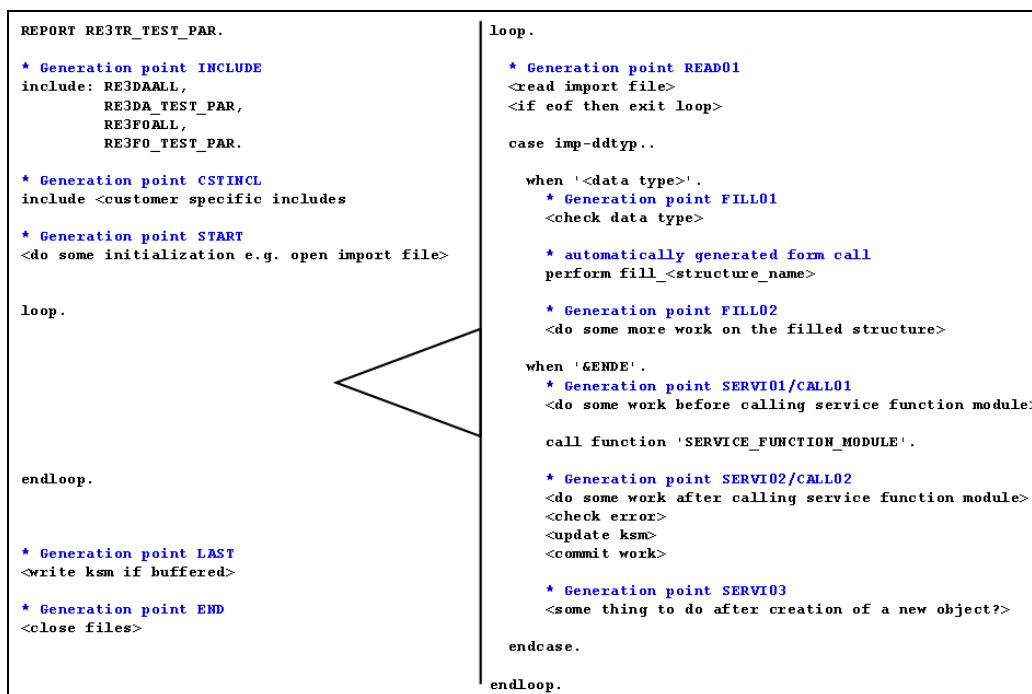


Figure 4-2: Layout of the Load Program

4.3 Custom Code in a Load Program

The IS Migration Workbench allows inclusion of own ABAP code in the following processing levels:

- Field (field rule ABAP and fixed value)
- Structure (pre- and post-processing)
- Load report (events)

The following chapter will explain in detail when and how to develop ABAP code in each of the processing levels. ABAP code that you enter cannot be checked immediately for syntax

errors. The syntactic correctness of the code is checked with the generation of the load report. This is due to a missing program environment and the usage of symbolic values. Instead of correcting the generated code, correct a code error in the source code. The load program generator overwrites the correction the next time the load report is generated.

Precede the comments in the code by an apostrophe instead of an asterisk. The load program generator indents the generated ABAP code (insertion of leading spaces) to make the generated code readable. As asterisks are only allowed in the first column of a report to indicate comment lines, an asterisk in the developed code will cause a syntax error during the generation of the load report.

4.3.1 Code on Field level

In a field with *ABAP* processing type, the contents of the corresponding field of the customer structure is processed before the respective field in the automation data structure is populated. You can determine the correct value for a field in the automation data structure using this field rule to access database tables and perform calculations or further processing. Figure 4-3 shows the *Field Rule Maintenance* screen.

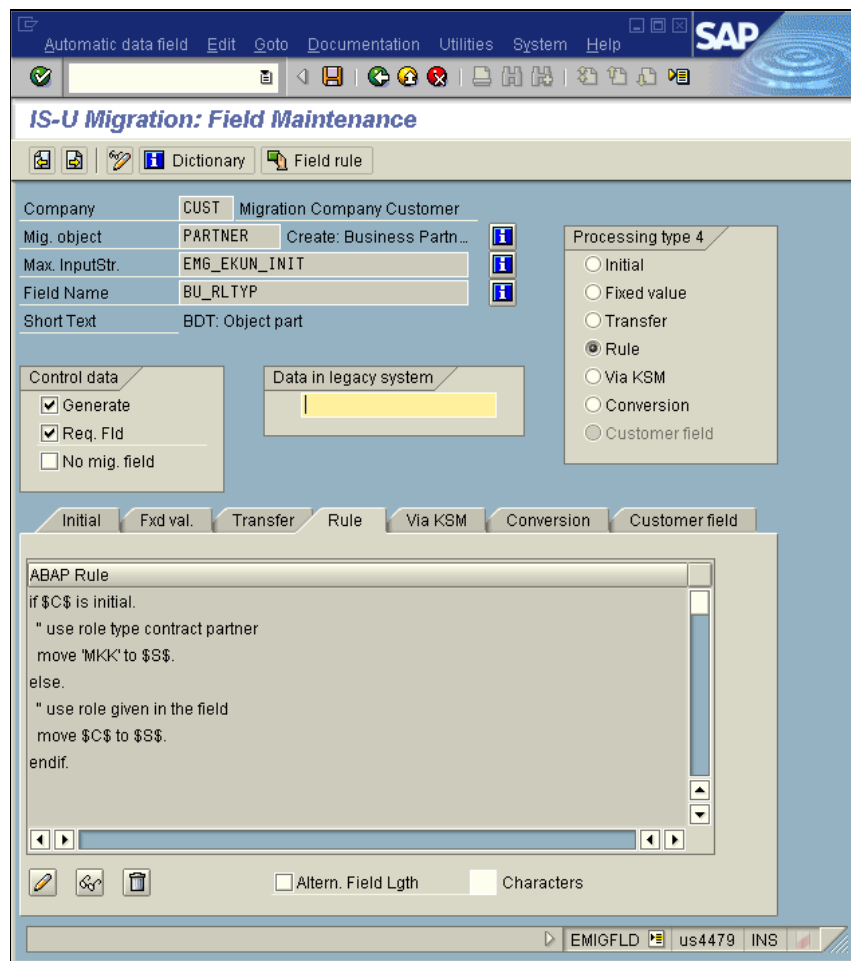


Figure 4-3 Field Maintenance Screen

To implement the code, choose the button in the **Rule** sub-screen and use the following symbolic variables:

- **\$C\$** is used as the placeholder for the field in the customer structure.
- **\$S\$** is used as the placeholder for the field in the automation data structure.

The load program generator replaces the symbolic variables using the correct field references to the fields in the customer and automation data structure when you generate the load report. The simplest code is \$\$\$ = \$\$\$\$. This moves the information from the field of the customer structure to the automation data structure without further processing. A more complex example is shown in Figure 4-4 ABAP Code in ABAP field Rule

. This code is implemented for the BU_RLTYP field of PARTNER migration object.

Code in ABAP Rule	Generated Code in Load Program
<pre> ABAP Rule if \$\$\$ is initial. " use role type contract partner move 'MKG' to \$\$\$\$. else. " use role given in the field move \$\$\$ to \$\$\$\$. endif. </pre>	<pre> * * Filling field EMG_EKUN_INIT-BU_RLTYP if z_EMG_EKUN_INIT-BU_RLTYP is initial. " use role type contract partner move 'MKG' to EMG_EKUN_INIT-BU_RLTYP. else. " use role given in the field move z_EMG_EKUN_INIT-BU_RLTYP to EMG_EKUN_INIT-BU_RLTYP. endif. translate EMG_EKUN_INIT-BU_RLTYP to upper case. </pre>

Figure 4-4 ABAP Code in ABAP field Rule

The load program generator replaces the symbolic variable \$\$\$ with the correct reference z_EMG_EKUN_INIT-BU_RLTYP and the symbolic value \$\$\$ with EMG_EKUN_INIT-BU_RLTYP.

You can change the length of the field in the customer structure. To do this, mark the **Altern. Field Lgth** field and enter the desired field length in the **Characters** field. Instead of defining the field in the customer structure with its original data dictionary attributed, the field is now defined as a character field with the entered number of characters. We recommend using this option when the field is defined in the data dictionary with the data type P (packed number) or I (integer).

The load program generator implements the execution of a conversion exit if one is defined in the dictionary for this field. The conversion exit will process the field in the automation structure.

It is technically possible to reference any field of the customer structure by preceding the field name by the symbolic variable \$\$\$- (for example, \$\$\$-fieldname) or any field of the automation data structure by preceding the field name by the symbolic variable \$\$\$- (for example, \$\$\$-fieldname). The load program generator replaces these symbols with the respective structure name instead of the complete field reference of the field that is currently processed. Even if it is possible to access all fields, the implemented ABAP code should be restricted to operate on only the field for which the field rule has been defined, except for custom fields (for more information, see chapter 4.4.1 *Additional Fields in the IS Migration Workbench*). This is because the load program generator determines the order of the execution of the field rules. There is no guarantee that a field in the automation structure has already been populated by another field rule when processing the ABAP code field rule.

4.3.2 Code on Structure Level

You can define own ABAP code at a structure level. The ABAP code entered under **Preprocessing** is carried out before processing any field rule of the actual structure. The ABAP code entered under **Postprocessing** is carried out after the field values have been processed, but before the required entry field checks. This makes cross-field processing possible at structure level.

An ABAP code can perform checks that involve more than one field of the customer or automation data structure and you cannot configure it on a field level. The code must be implemented using symbolic variables:

- **\$C\$** is used as the placeholder for the customer structure.
- **\$S\$** is used as the placeholder for the automation data structure.

The load program generator replaces the symbolic variables with the correct structure references to the customer structure and automation data structure when generating the load report. Figure 4-5 shows the structure maintenance screen with the sub screens **Preprocessing** and **Postprocessing**.

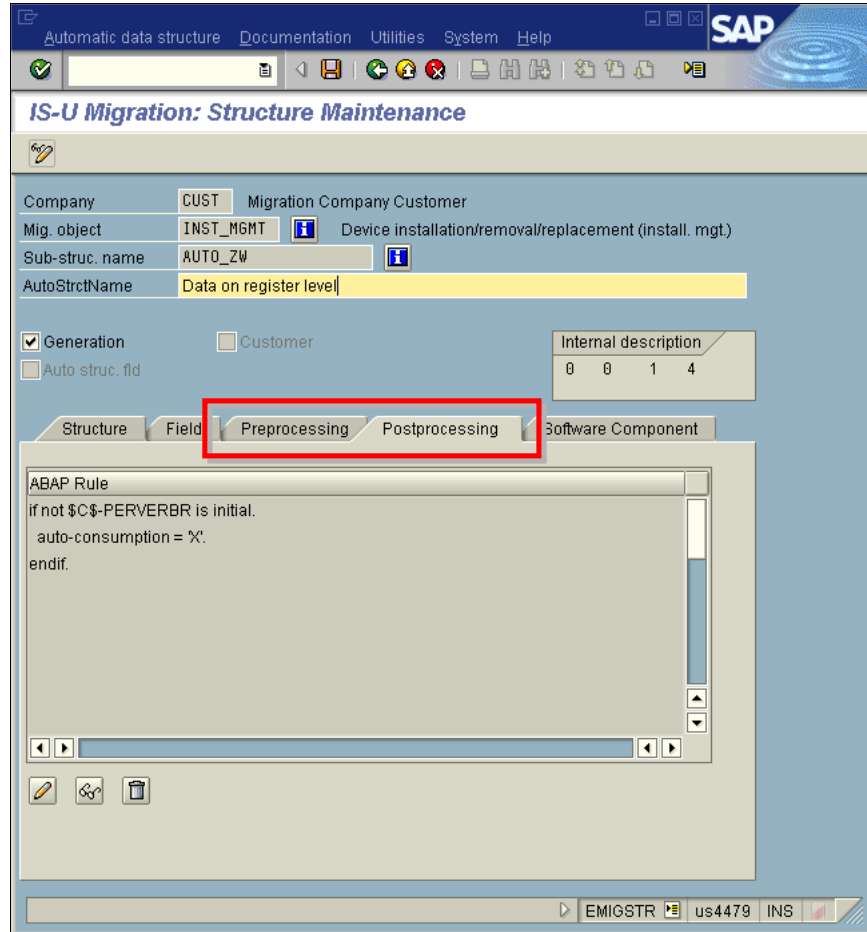


Figure 4-5: Structure Maintenance Screen

You can also manipulate data directly in the automation data structure while processing the ABAP code for postprocessing. The structures of the automation data can be addressed by the prefix **AUTO** (for example, **AUTO-<structure_name>**). This is not common, but it is possible. Figure 4-5 shows this. The post processing code changes an indicator in the auto data depending on the field content (here it indicates that a period consumption value has been transferred).

4.3.3 Code on Report Level

The load program generator generated the load program mainly based on events (for more information, see chapter 4.2 *Layout of the Load Program*). When you do not create a migration object as a copy of an existing migration object, the events of the **ACCOUNT** migration object in migration company SAP are used as a template during the creation process. Only when you create a hyper migration object (for more information, see chapter 3.5 *Hyper Object*), the special events of **HYPER** migration object in migration company SAP are copied into the new hyper object. The standard events should neither be changed nor

removed to ensure a proper execution of the load program. Figure 4-6 shows the *Event Maintenance* screen. Events are bound to a load program and cannot be shared between migration objects.

You can create the following different kinds of events:

- **Perform** (Perform a Subroutine)

The load report calls the entered subroutine using a `PERFORM` statement. You must define a parameter variant for the parameters to be passed to the form routine. Incorporate the subroutine within a standard or a user defined include program, which must be included with another event.

- **Function** (Perform a Function Module)

A call of the entered subroutine is included in the load report and is called with a `CALL FUNCTION` statement. The parameters to be passed to the function module have to be defined by a parameter variant.

- **Include** (Insert an Include Program)

An include program is included in the load program by an `INCLUDE` statement.

- **Code** (Insert ABAP Statements)

ABAP statements are copied into the load program. You have to give the ABAP code a name. This is entered in an editor that is called after choosing the *Code* button.

Figure 4-6 Event Maintenance Screen

It is possible to add a post processing code to the generated event in the load program. This is mainly used to add error handling (for example, to the call of function module). To do this, you must enter a name in the **Code** field on the *Event Maintenance* screen, and call the code editor by choosing the *Code* button. The developed ABAP code can be used in different events for the load program of a migration object by using its name as a reference.



The load program generator includes events according to their sort order per generation point, and only, if the event has been activated by marking the **Activate Event** field on the *Event Maintenance* screen (see the red box in Figure 4-6).

The SAP standard includes two subroutines that you can use for special functions in the events.

- **Perform BREAK**

This event can be added to at any generation point and processes a break-point within the load program.

- **Perform DUMMY and Perform CODE**

These dummy subroutines still exist for compatibility reasons. They have the same function as CODE, and they are used only to ensure backward compatibility for existing events. The new event type CODE has replaced them.

Figure 4-7 shows, how event in Figure 4-6 is implemented in the load program of the ACCOUNT migration object.

```
perform CHECK_COMMIT.
perform DEAKTIVATE_PARAMETER.
perform HOLD_TIME
  USING 'start:'.

CALL FUNCTION 'ISU_M_KSV_PUFFER'
  EXPORTING
    X_PUFFERS           = SPACE
    X_PUFFERI           = 'X'
    X_ANZAHLI           = ANZ_PUFF
    " X_USE_TEMKSV_BUFFER = 'X'
  TABLES
    XT_TEMPUP          = ITEMPUF
.
if sy-subrc ne 0.
  exit.
endif.
*$$$ end of section 2
*$$$ start of section 3

do.
```

Figure 4-7 Generated Event in the Load Program

4.3.4 Error Handling in Custom Code

A special message handling is implemented in the load program to ensure that an error message does not cause a cancellation of the load program. The error messages are collected in the error (application) log for later evaluation. In the load program, some global variables and macros are available to signal problems. These cancel the processing of the present data object to be migrated. In the case of an error, the own code should signal an error in the following steps:

- **Raise error message**

The `MAC_MSG_PUTX` macro is used to raise the error message to be saved in the error log. The macro requires the following parameters:

- Message type (must be 'E')
- Message number
- Message-class
- Message variables 1 to 4
- Exception (must be space)

- **Signal an error**

The `MAC_JUMP` macro is used to signal an error to the load program. When an error is signaled in this way, the service function module is not called. The mandatory parameter of this macro is '6'.

Figure 4-8 shows an example of how to implement error handling on the structure level (postprocessing). Use the generic message class EM and message number 000 to pass a free text to the error log while processing an FI-CA document.

```
" cross field check for document line items
IF NOT $$$-invoicing_party IS INITIAL AND
  $$$-vtref IS INITIAL.
```

```

" store error message
mac_msg_putx 'E' '000'
           'EM' 'Missing contract reference'
           'for service provider' $$-invoicing_part
           space space.
" raise error and signal cancellation
mac_jump 6.
ENDIF.

```

Figure 4-8: Example of Error Handling in the Custom Code of a Load Program

If an event has been implemented that calls a function module, any error raised within the function module must be treated appropriately. The simplest way to do this is by adding a postprocessing code to the event as described in the previous chapter. Figure 4-9 shows the implementation of a generic code.

```

" error occurred?
IF sy-subrc NE 0.
" store error message
mac_msg_putx 'E' sy-msgid sy-msgno
           sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4
           space.
" raise error and signal cancellation
mac_jump 6.
ENDIF.

```

Figure 4-9: Example of a Postprocessing Code a Call of a Function Module

4.4 Customer-Defined or Additional Fields

There are two kinds of customer fields:

- Fields as part of the customer structure of the IS Migration Workbench
- Fields as part of an enhancement of the standard structure or table

Customer fields are defined in the IS Migration Workbench and are visible and accessible only in the load program, whereas the fields of an enhancement of the solution can be processed not just in the IS Migration Workbench but in either dialog transactions and/or batch processes.

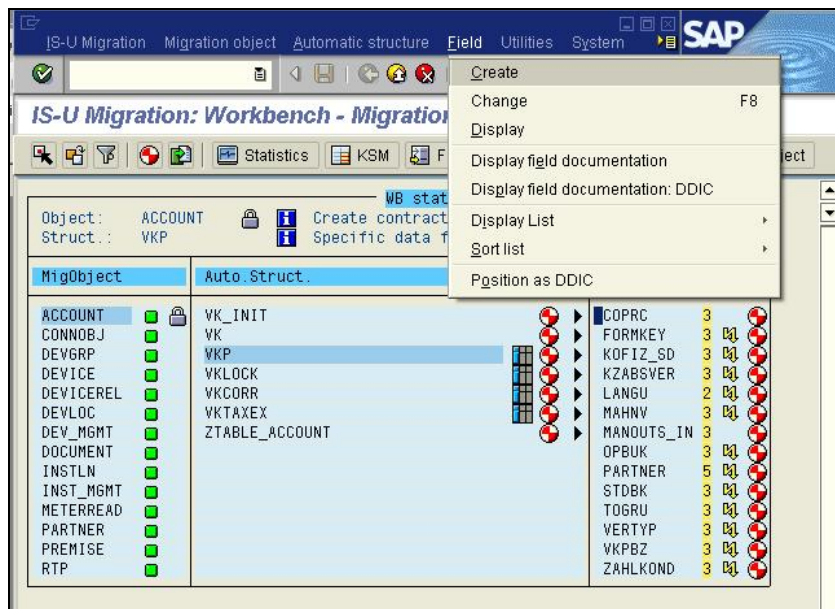
4.4.1 Additional Fields in the IS Migration Workbench

Figure 4-10 shows how to define an additional field in the customer structure of the ACCOUNT migration objects in the IS Migration Workbench. Each time you add a new field, the layout of the related structure in the import file changes accordingly. For more information about the *Customer Field* field rule, see chapter 2.4.7 *Field Rule Customer Field*.

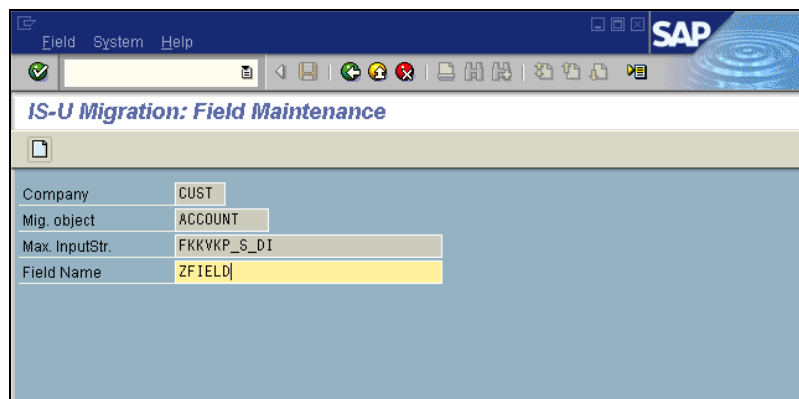
1. Double-click the ACCOUNT migration object

MigObject	Auto Struct.	Field
ACCOUNT	VK_INIT	
CONNOBJ	VK	
DEVGRP	VKP	
DEVICE	VKLOCK	
DEICEREL	VKCORR	
DEVLOC	VKTAXEX	
DEV_MGMT		

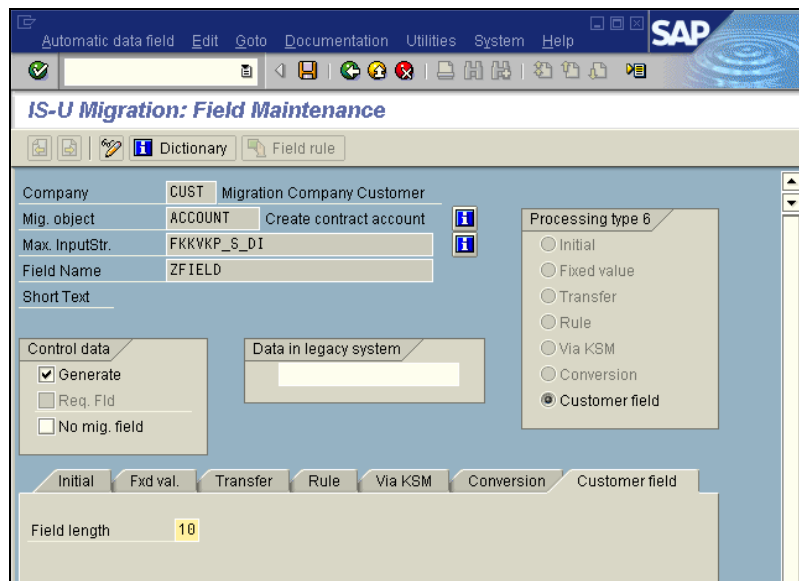
2. Double-click the relevant structure to mark it, choose *Field* → *Create*



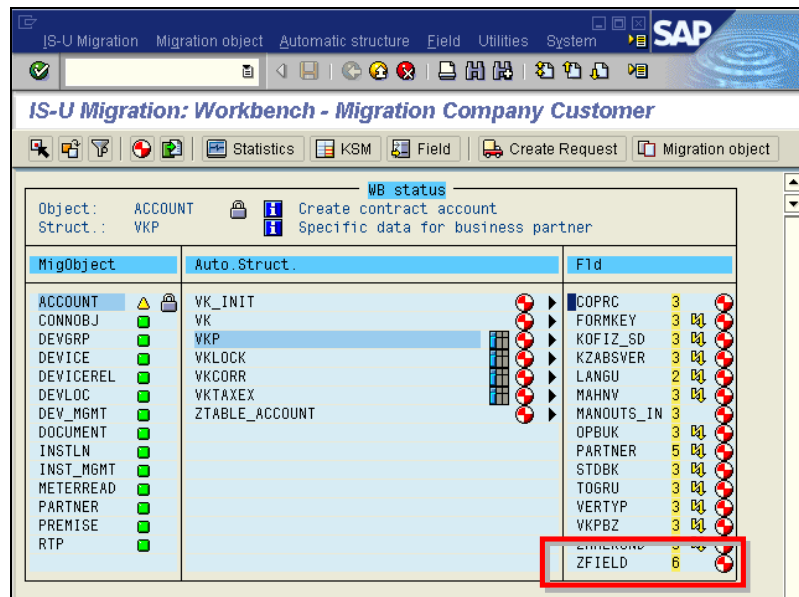
3. Enter the name of the new field and choose *Create*



4. Mark the *Generate* field and enter the length of the new field. The *Customer field* rule is selected automatically.



5. Choose Save and return to the main screen to display the added field.



6. Now you can use a field rule to access the newly created field.

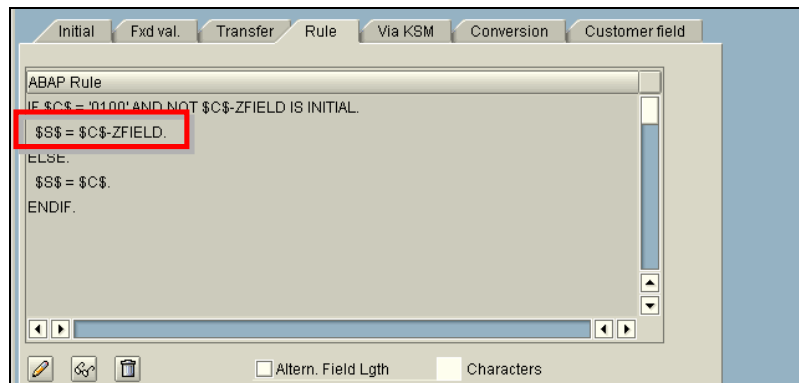


Figure 4-10: Adding a Field to the Customer Structure

4.4.2 Enhancement Fields in the Solution

Fields are part of the solution and have been included in a customer-include. They become part of the ABAP dictionary (DD) and a part of the automation data. The IS Migration Workbench recognizes automatically a new field in the automation structure during a copy of a migration object. If you have added a field after the migration object has already been copied, the field has to be added manually using the *Compare with DDIC* function. Double-click the related structure to mark it and choose *Auto structure* → *Compare with DDIC*.

Usually, it is not enough just to include the field in a customer-include. In most cases, it is mandatory implement either user-exits or events relevant for direct input in order to process the customer fields. This implementation is not part of the migration Customizing, it is part of the development of the solution.

The following chapters show examples of what is required when you are enhancing the solution with new fields.

4.4.3 Enhancement Fields for Technical Master Data (SAP for Utilities)

There are two relevant enhancements (transaction CMOD) to process a new field for a connection object:

- Enhancement ECIIFLOT to add the new fields in the customer-include CI_IFLOT
- Enhancement ES550001 (component EXIT_SAPLES55_008) to process the fields during direct input.

Both of the enhancements have to be implemented in order to save the contents of the new fields, for example, in the EHAU database table. Figure 4-11 shows a code snippet, that transfers the contents of the custom fields in user-exit 008 during direct input.

```
*& Include ZXS55U08
*&-----*
DATA: lv_ci_iflot TYPE ci_iflot.

MOVE-CORRESPONDING x_ehau TO lv_ci_iflot.
MOVE-CORRESPONDING lv_ci_iflot TO xy_sub_obj-ehau.
```

Figure 4-11: Example of an Implementation of Enhancement ES550001

You must apply a similar solution for many of the enhancements in SAP for Utilities. Implement, for example, component EXIT_SAPLES65_007 in enhancement ES650001 to process custom fields for device locations and component EXIT_SAPLES60_007 in enhancement EMDP0001 to process custom fields in premises.

4.4.4 Enhancement Fields for a Point of Delivery (SAP for Utilities)

A deregulated point of delivery (PoD) is created automatically when creating an installation with the INSTLN migration object. The automation data structure of the INSTLN migration object does not include the customer-include CI_EUIHEAD in which you added the customer fields. Therefore, a direct migration of the customer field with the INSTLN migration object is not possible.

Execute the migration of customer fields for a PoD in the following way:

- Create the point of delivery (POD migration object)
 - Transfer the customer fields in the HEADER structure
 - Transfer an external unique identification in the EXT_UI field of the UIEXT structure.
- Create the installation (INSTLN migration object) and link the created PoD to the installation in transferring external unique identification in the EXT_UI field of the POD structure, you assigned to the created PoD in the previous step.

Alternatively, you can create the installation with the INSTLN migration object leaving the customer fields initial, and then populate the customer fields in the existing PoD with the PODCHANGE migration object.

One more alternative is to taking advantage of the method CHANGE_AUTODATA of the BAdI (business add-in) ISU_EDM_POD_CUSTOMER. As explained above, you cannot pass the information for the customer fields directly through the automation data to the application due to the missing customer-include in the automation data structure of the INSTLN migration object. Nevertheless, you can implement the migration of the customer fields of a PoD also in the following way:

- Add the customer fields of the customer-include as customer defined fields with the *Customer Field* field rule to the DATA structure.
- Add per customer field the code of Figure 4-12 to the postprocessing code of the DATA structure.

```

call function 'ISU_MIG_PARAMETER_SET'
  exporting
    x_element = '<name_of_customer_field>'
    x_field   = '$C$-<name_of_customer_field>'.

```

Figure 4-12 Postprocessing Code for DATA structure

- o Implement the method `CHANGE_AUTODATA` of the BAdI `ISU_EDM_POD_CUSTOMER` based on the ABAP code as shown in Figure 4-13.

```

METHOD if_ex_isu_edm_pod_customer~change_autodata.
* local data definition
DATA: lv_active TYPE c,
      lv_object TYPE temob-object.
DATA: lv_ci_euihead TYPE ci_euihead.

* check whether data migration is active
CALL FUNCTION 'ISU_MIG_STATUS'
  IMPORTING
    y_active = lv_active
    y_object = lv_object.
IF lv_active = 'X' AND
   lv_object = 'INSTLN'.

* get customer field
CALL FUNCTION 'ISU_MIG_PARAMETER_GET'
  exporting
    x_element = '<name_of_customer_field>'
  importing
    x_field   = lv_ci_euihead-<name_of_customer_field>.

* move customer field to auto data
MOVE-CORRESPONDING lv_ci_euihead TO ch_autodata.

ENDIF.
ENDMETHOD

```

Figure 4-13 Method `CHANGE_AUTODATA` of BAdI `ISU_EDM_POD_CUSTOMER`

In the postprocessing code of structure `DATA`, the field content of the customer field is moved to a container, picked up from the container in the method `CHANGE_AUTODATA` and moved finally to the auto data of the BAdI. The `ISU_MIG_STATUS` function module returns in the `Y_ACTIVE` parameter the information whether the application has been called from a load program of the IS Migration Workbench. The name of the migration object is returned in the `Y_OBJECT` parameter. However, deactivate the BAdI once the data migration finished because the BAdI is not required after the go-live.



The third alternative is not the best way to implement the migration of customer fields for a PoD, because the presented implementation is not following the standard programming guidelines. However, considering the aggravating circumstances not having the customer-include available in the automation data structure of the `INSTLN` migration object, this alternative still might be the most efficient way to migrate customer fields for a PoD.

4.4.5 Enhancement Fields for Business Partners

The Easy Enhancement Workbench (EEW) allows you to extend the data model of the Business Partner easily by adding new fields or tables, and to integrate them into the business process.

The Easy Enhancement Workbench is a development tool with which SAP applications for business partner, business partner relationship, and others, can be extended in a simple manner. The customer enhancements to a Business Object are defined via wizards. The

Workbench then does all development work for the user; databank tables, screens, and application logic are created automatically. Finally, the customer enhancement is included in the SAP standard. The type of extension is predefined. In most cases, the customer is offered the possibility to add user-defined databank tables or fields. The Easy Enhancement Workbench (transaction `EEWB`) enables users without ABAP knowledge to extend the SAP standard.



The SAP notes 510095 and 119805 give directions for an implementation of enhancements with BDT if the Easy Enhancement Workbench is not used.

4.4.6 Enhancement Fields for Contract Accounts

Use the Business Data Toolset (BDT) to implement the required enhancements. Ensure that at least one event at the DINP2 generation point is implemented. For more information, see the developer handbook of the Business Data Toolset in the online documentation. The code snippet in Figure 9-2 can be used as a basis for an own implementation.

5 Migrating Custom Tables (z-Table)

5.1 Motivation

In almost any data migration, you must populate custom database tables with data from the legacy system. Often custom database tables are used to store historical data of the legacy system, which does not fit into the SAP data model. Sometimes it cannot even be cleansed sufficiently prior to going live to be passed through the SAP application. More often custom tables are part of the implementation and additional data is stored for one more business objects. In either case, these custom database tables have to be populated during data migration.

There are two ways to migrate data into the custom table efficiently using the IS Migration Workbench:

- Using an existing (standard) migration object
- Using an own migration object

A custom program is an option but this not the best solution in most cases. This is because in many cases, fields of a custom database table contain a reference to existing business objects. Additional development would be required to ensure the replacement of the oldkey with the correct newkey. In addition, it is not only preferable using the same file format for all data to be migrated into the SAP system but also taking advantage of the functions of the IS Migration Workbench in error handling, statistics, and parallel load.

5.2 Migrating Custom Tables Using an Existing Migration Object

You can use an already existing migration object to migrate data to a custom table. The necessary data is transferred in addition to the data that is transferred for the business object to be migrated. For this, one more structure is added to the migration object. This structure allows you to transfer data with the necessary information to populate the custom table while migrating the migration object. The advantage compared to a further migration object is that no additional migration step needs to be implemented to populate the custom table. Figure 5-1 shows the procedure to add an additional structure to the ACCOUNT migration object.

1. Double-click the ACCOUNT migration object

MigObject	Auto.Struct.	Fld
ACCOUNT	VK_INIT	
CONNOBJ	VK	
DEVGRP	VKP	
DEVICE	VKLOCK	
DEVICEREL	VKCORR	
DEVLOC	VKTAXEX	
DEV_MGMT		

2. Choose *Automation Structure* → *Create*
3. Enter the name of the custom table and choose *Create*.

Substructure System Help

IS-U Migration: Structure Maintenance

Company: CUST

Mig. object: ACCOUNT

Sub-struct. name: ZTABLE_ACCOUNT

4. Enter the description for the structure in the **AutoStrctName** field. Enter the name of the custom structure as **Max. Input Structure**. Mark the checkbox **Customer** to indicate that the new structure is not part of the automation structure, and enter a unique **Record Data type** (it must be unique within the ACCOUNT migration object).

Automatic data structure Documentation Utilities(M) System Help

IS-U Migration: Structure Maintenance

Company: CUST Migration Company Customer

Mig. object: ACCOUNT Create contract account

Sub-struct. name: ZTABLEACCOUNT

AutoStrctName: Structure to migrate additional information

☒ Generation ☒ Customer

☐ Auto struc. fld

Internal description: 0 0 1 7

Structure Field Preprocessing Postprocessing

Max. InputStr.: ZTABLEACCOUNT

Rec data type: ZTAB ☐ Multiple legacy keys

☐ Multiple

Variable type: " "

5. Choose **Save** and return to the main screen to display the added structure.

WB status

Object: ACCOUNT Create contract account

MigObject	Auto.Struct.	Fld
ACCOUNT	VK_INIT	
CONNOBJ	VK	
DEVGRP	VKP	
DEVICE	VKLOCK	
DEVICEREL	VKCORR	
DEVLOC	VKTAXEX	
DEV_MGMT	ZTABLE_ACCOUNT	
INSTLN		
INST_MGMT		
METERREAD		
PARTNER		
PREMISE		
RTP		

6. Double-click the newly created structure. Choose **Automation Structure** → **Compare with DDIC**. After confirming the popup displaying the newly added fields, the fields of the custom table are also displayed in the IS Migration Workbench.

WB status		
Object:	ACCOUNT	Create contract account
Struct.:	ZTABLE_ACCOUNT	Structure to migrate additional information
MigObject	Auto.Struct.	Fld
ACCOUNT	VK_INIT	GPART 1
CONNOBJ	VK	MANDT 1
DEVGRP	VKP	VKONT 1
DEVICE	VKLOCK	ZCUSTOM1 1
DEVICEREL	VKCORR	
DEVLOC	VKTAXEX	
DEV_MGMT	ZTABLE_ACCOUNT	
INSTLN		
INST_MGMT		
METERREAD		
PARTNER		
PREMISE		
RTP		

7. Maintain the field rules for the fields of the added created structure.

WB status		
Object:	ACCOUNT	Create contract account
Struct.:	ZTABLE_ACCOUNT	Structure to migrate additional information
MigObject	Auto.Struct.	Fld
ACCOUNT	VK_INIT	GPART 5
CONNOBJ	VK	MANDT 1
DEVGRP	VKP	VKONT 1
DEVICE	VKLOCK	ZCUSTOM1 3
DEVICEREL	VKCORR	
DEVLOC	VKTAXEX	
DEV_MGMT	ZTABLE_ACCOUNT	
INSTLN		
INST_MGMT		
METERREAD		
PARTNER		
PREMISE		
RTP		

Figure 5-1: Procedure to Add an Additional Structure to a Migration Object

After the field Customizing has been completed, an event has to be generated in the load report in order to populate the custom table. The event can be created, for example, at generation point CALL02 with the ABAP code shown in Figure 5-2. The procedure for the generation of a new event in a migration object is described in 4.3.3 *Code on Report*.

```
" populate customer table ZTABLE_ACCOUNT
IF sy-subrc = 0 AND db_update = 'X'.
  " return value from service function module
  ztable_account-vkont = new_acc-vk-vkont.
  INSERT ztable_account.
ENDIF.
```

Figure 5-2: ABAP Code at Generation Point CALL02 to Update Custom Table

5.3 Migrating Custom Tables Using an Own Migration Object

You can easily develop an own migration object for the migration of a custom table. Figure 9-3 shows the ABAP code of a function module that you can use as a template for a service function module in a BAPI migration object. Either the function module does a direct insert in the `ZTABLE` database table or the insert is deferred until a `COMMIT WORK` is executed. The procedure for the generation of the new BAPI migration object is described in chapter 3.4.2 *Generation of a BAPI Migration Object*.

After the generation of the BAPI migration object, the necessary field Customizing can be done. Figure 5-3 shows the finished configuration in the IS Migration Workbench.

WB status			
Object:	ZTABLE	Data Migration Databas Table ZTABLE	
Struct.:	XT_ZTABLE_TAB	Project specific database table	
MigObject	Auto.Struct.	Fld	
ZTABLE	IMPORT_FIELDS	LFDNR	3
	RETURN	MANDT	1
	XT_ZTABLE_TAB	PARTNER	3
		ZCUSTOM1	3
		ZCUSTOM2	3
		ZCUSTOM3	3

Figure 5-3: Automation Structure of the Generated Migration Object

6 Migration Strategies (SAP for Utilities)

6.1 Motivation

It is imperative while designing a migration strategy, not to focus on only single business objects and their counterparts as migration objects in the IS Migration Workbench. You must also analyze the business processes, which are implemented in the SAP system. Data migration is not only used for migrating data from the legacy systems to the SAP system. You have to understand data migration as a transfer of business processes, which are continued in the SAP system after the go-live, based on the migrated master and transaction data. The following chapters highlight the relationship between the migration objects and describe some important strategies.

6.2 Installation Structure History

One of the most fundamental decisions in an implementation project is about the migration of installation structure history. The basic question is, to which extend historical data of installations, device installations, historical contracts, and billing documents have to be migrated from the legacy system to the SAP system.

You must distinguish between the time dependencies of data objects:

- The data for devices (DEVICE migration object), connection objects (CONNOBJ migration object), premises (PREMISE migration object), device location (DEVLOC migration object), and installations (INSTLN migration object) are migrated as they are in the legacy system. The installation must exist at the day of the first installation; fact (FACTS migration object) has to be created for this installation.
- The dates for the device installations (INST_MGMT migration object), contracts (MOVE_IN migration object), and for further objects depends on the migration strategy to be implemented.

The following chapters describe the various standard scenarios starting from the simplest scenario to the most complex one.

6.2.1 Standard Scenario (No Installation Structure History)

Figure 6-1 describes the standard migration scenario. The guiding principle is to avoid the migration of historical data.

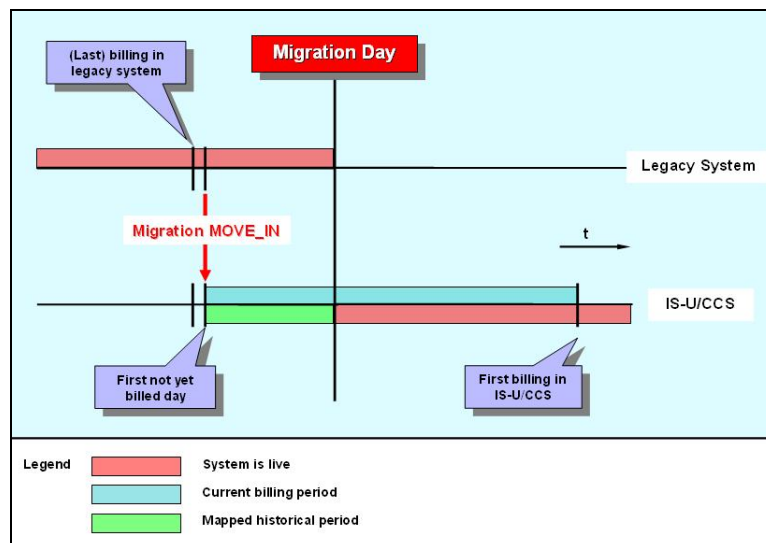


Figure 6-1: Standard Scenario

In this scenario, you must apply the following migration rules:

- The last legacy billing date of the installation in the legacy system determines the **move-in date** and **device installation date**. Both are set to the first day not yet billed.
- The **installation from-date** should be before the day you create the first installation fact, and before the day the devices will be installed billing related.
- Installation facts (for example, consumption history) can be migrated for any date that is valid for the installation
- Meter readings can be migrated for any day between device installation and the current day of migration

It must be part of the go-live strategy to suspend any changes during the final weeks before the go-live date. A migration of any device changes in an installation leads to more complex migration scenario as described in the next chapter.

6.2.2 Extended Device History

The Figure 6-2 describes the scenario of the migration of an extended device history. The guiding principle is the need for a migration of meter reading history. For this, you have the following options:

- Migration of device history for a defined period (for example, 6 months)
- Migration of device history for the device which is installed at the time of data migration

The first case will cause many problems because the migration of device replacements and device modifications must be migrated too. This can be difficult because device replacements, device relationships, device modification, and the respective meter reading reasons have to be migrated. All device related activities have to be tracked cross-installation because these activities have to be migrated in the correct historical order. To do this, use hyper migration objects (for more information, see chapter 3.5 *Hyper Object*). The second case will not cause major problems.

In both cases, during the migration of a billing related installation of device, the application will automatically create the meter reading reason 06 for a move-in (for more information, see chapter 7.9 *FAQ Meter Reading*). The automatic creation of the meter reading reason move-in has to be suppressed if the device installation date is not the same as the move-in date of the contract.

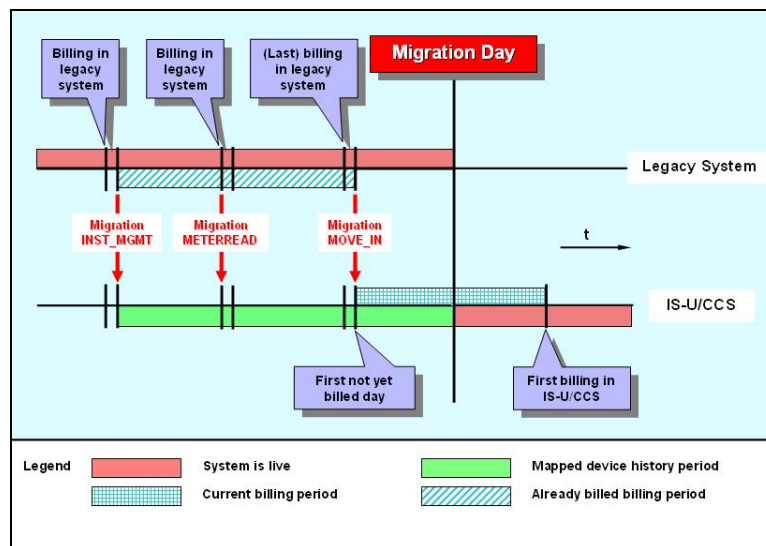


Figure 6-2: Extended Device History

Apply the following migration rules:

- The **move-in date** is determined by the last legacy billing date of the installation in the legacy system. It is set to the first day not yet billed.
- The **device installation date** is determined from the legacy system by the any past legacy billing date.
- Changes in the allocation of devices to device location or installation, as well as, any change in the rate data or device or register relationships have to be migrated in **chronological order**.
- The **installation from-date** should be before the day that you create the first installation fact, and before the day the devices will be installed billing related.
- Installation facts (for example, consumption history) can be migrated for any date that is valid for the installation.
- Meter readings have to be imported for the period between device installation and the migration day.

6.2.3 Extended Billing History

Figure 6-3 describes the scenario of the migration of an extended billing history. The guiding principle is the need for a migration of billing documents. Implement an extended billing history for the following reasons:

- Legal requirements
- The implementation of a cancel and re-invoicing process (correction of an incorrect bill) in the SAP system
- Display of historical billing information in the customer facts sheet (call center)

There are the following alternatives to migrate historical billing documents:

- Migrate the billing history with real billing documents (BILLDOC migration object)
- Migrate the billing history with EBF billing documents (BILLDOCEBF migration object)

In the first case, the extended device history also has to be migrated with all problems already described. In the second case, the migration approach can be simplified. For more information on migration of historical billing documents, see chapter 6.3 *Historical Billing Document*.

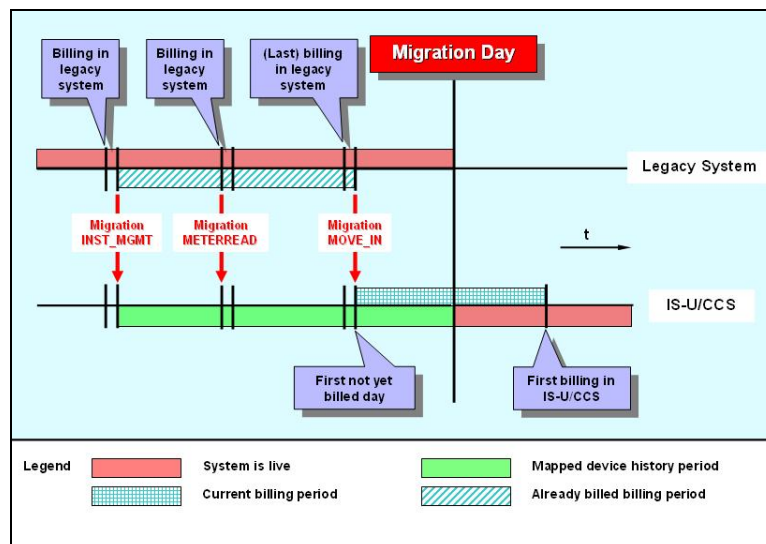


Figure 6-3: Extended Billing History

In this scenario, apply the following migration rules:

- All the rules that are valid for the migration of an extended device history (if real billing documents are migrated with the BILLDOC migration object)
- The billing date of the billing document must match the legacy billing date without overlaps or gaps.
- The move-in date has to precede or match the legacy billing date of the oldest migrated billing document. In the latter case, the move-in date sets the start of billing period for the period that is ended by the first migrated BILLDOC.
- The last migrated billing document represents the last legacy bill and sets the start date of first billing period in the SAP system.
- The billing documents have to be created for at least the last day of the corresponding legacy bill.

6.3 Historical Billing Document

6.3.1 Migration Objects Relevant for Migrating Billing Documents

The following migration objects support the migration of historical billing documents.

- Use the BILLDOC Migration object to create billing documents for installations. A requirement is a complete and consistent installation structure.
- Use the CONSHIST migration object to create core-billing documents for mapping the consumption history (DBERCHV database table). Not many checks are carried out when the consumption history is created. This means that it is not necessary to migrate the historical installation structure.
- Use the BILLDOCEBF migration object to create EBF billing documents. Individual correction methods are available for making corrections to these billing documents. You make the basic settings (activation of functions) for the easy bill correction framework. After a successful go-live, it is possible to migrate EBF billing documents.
- Use the MOVE_IN_H migration object to create historical move-ins and move-outs. This allows you to create billing documents for inactive contracts. Generally, it is possible to do this after the go-live.
- If the billing schemas are very close to the billing in the legacy system, it is possible to execute the billing in the SAP system once the installation structure has been migrated successfully. To do this, use the BILLTRIGCR migration object to create a billing order. You can use the BILLTRIGG migration object to execute billing in a subsequent step.
- A popular requirement for utility companies is to have an automated bill correction scenario available within SAP for Utilities. Whilst this is achieved through the EABICO transaction for billing documents created by the processes in SAP for Utilities, the difficulty arises for billing documents that have not been created in the SAP system (if they exist in the Legacy environment), or if they are no longer available in the SAP database as a result of archiving activity.
- The only way to make legacy billing documents available for a correction is to migrate them into the SAP database using the BILLDOC migration object. A full historical migration of all the data needed to do a full automatic bill correct in the SAP system is nearly impossible. The reason for this is that the data structures of the legacy system are likely to be very different to those in SAP for Utilities. Therefore, it would be very difficult to match the data available with the SAP database fields to create billing documents, invoice documents, and print documents, which are all consistent. Today there is no option available for migrating print document.

You must have consistent data available for a reversal and an automatic re-bill process. You must also have all historical system data available such as rates, prices, schemas, and taxes. Therefore, a full historical data migration would be required in order to fulfil the re-

bill requirements of the application. This would impose a high risk on the project and would be very costly. Figure 6-4 compares these alternatives.

SAP developed the *Easy Bill-Correction Framework for Migration* (EBF) that allows the customers making the required bill corrections based on minimal data migration effort.

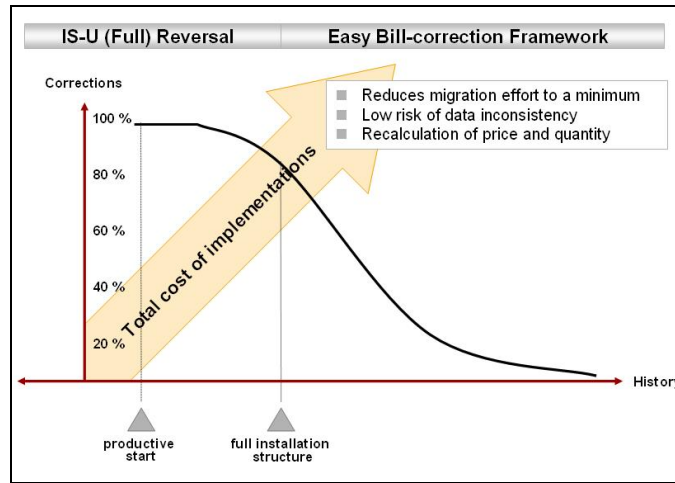


Figure 6-4: Total Costs of the Implementation

6.3.2 Easy Bill Correction Framework (EBF)

6.3.2.1 Introduction

The Easy Bill-correction Framework (EBF) is a toolbox for simple corrections or recalculations of billing document lines coming from a non-operational database, without any installation structure. EBF is a more automatic way to create a manual billing document based on reduced information.

EBF is able to perform the following operations:

- Migrate billing documents from the legacy system in a non-operational customer database (EBF data pool), using the BILLDOCEBF migration object.
- Use customer-specific correction methods on line item level
- Create a manual billing document for single corrections simply.

EBF cannot perform the following operations:

- Create a manual billing document from the operational SAP database
- Handle any installation-structures or facts

This new form of document line item correction is intended for billing documents that have been migrated from a source system, or for billing document line items that were created in the SAP system but have already been archived. It is similar to an adjustment reversal in that the original document is not reversed.

The new function consists of the following elements:

- **Data Pool**

The data pool is a logical view of the correctable billing documents, whose header data exists in the ERCH operational table. For migrated billing documents, it is now possible to define a specific storage location outside of the operational database table. Only read-only access to these reduced billing document line items is granted. You can make corrections only by using the new correction framework. You can configure the application area to specify whether you can use the new correction function only for migrated billing document line items or whether you can also use it for line items that have already been archived.

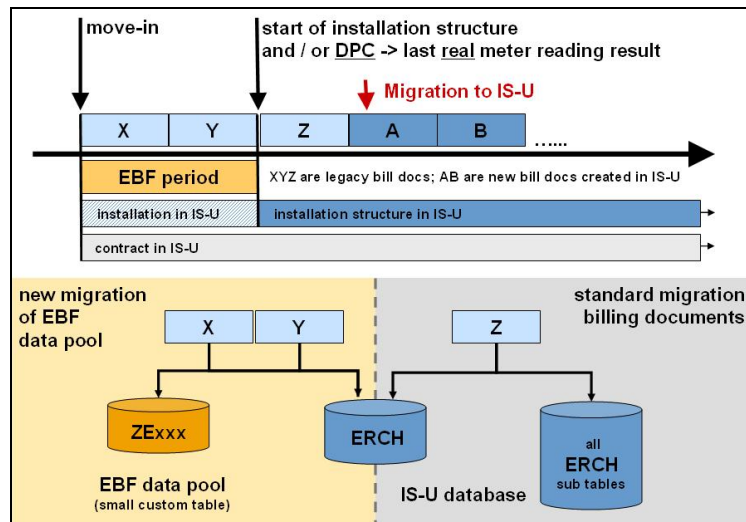


Figure 6-5: EBF Data Pool

- **Correction Framework**

The correction framework enables you to correct the document line items of a billing document from the data pool using correction methods, which you can define individually. The result of the correction process is a manual billing document, which you can further process using the normal standard functions. You can use business add-ins (BAIs) to further automate the correction process.

- **Effects on Data Migration**

If you want to use the new function to correct documents from the legacy system, use the new BILLDOCEBF migration object to migrate the data. This object has a simplified interface with reduced document data and it saves the data in the defined storage location in the data pool. As a result, it reduces the efforts of migration and less memory is required in the database compared to a full migration. The alternative BILLDOC migration object is still available without restrictions. It is used in particular for backbilling and dynamic period control.

6.3.2.2 Migration Scenarios

Figure 6-6 shows four of the most common scenarios, which you can implement. It is possible to mix different migration strategies.

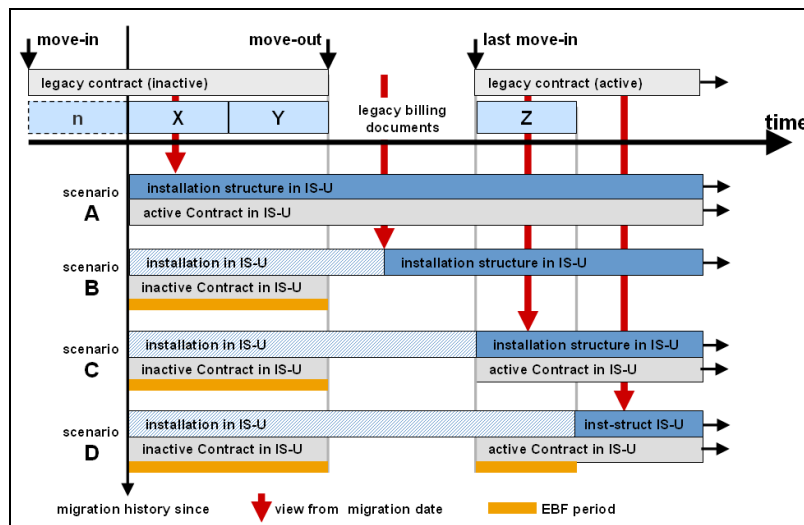


Figure 6-6: Migration Scenarios

The availability of historical data in the SAP system determines the options for a migration of historical billing documents. The *migration date* in the figure defines the date all the processes will work without limitation, due to the available of only reduced information. The *migration history since* in the figure is an individual date for each contract and has to be defined in each project. Billing periods to be migrated as EBF billing documents are marked with an orange line.

The following chapters provide generic information about data migration approaches. Minimally, a business partner, a contract account, and a contract is necessary in the SAP system to store the historical billing data stored in the EBF data pool.

6.3.2.3 Migration Scenario A

This scenario describes the standard migration scenario. An active contract exists in the legacy with a move-in date starting with the first not yet billed date. There is no option to migrate any historical data. The standard migration objects create an installation structure and a move-in, starting with the first billing period.

6.3.2.4 Migration Scenario B

In the legacy system, one (or more) inactive contracts and their corresponding billing documents exist. For the inactive contracts, the EBF billing documents can be migrated. The installation structure can be migrated starting with the migration key date without an extended device history.

6.3.2.5 Migration Scenario C

Migration scenario C is a combination of scenarios A and B. In the legacy system, an inactive contract with its legacy billing documents and an active contract without billing documents exists. No billing documents are available for the active contract due to the migration key date. For the inactive contract, it is sufficient to migrate EBF billing documents without an installation structure history.

For the active contract, an installation with installation structure history is required in the SAP system to handle the new billing document. From the migration point of view, it is necessary to use the standard migration objects to create an installation structure history and a contract.

6.3.2.6 Migration Scenario D

Scenario D is an extension of scenario C. However, a billing document also exists for the active contract in the legacy system. The closed contract situation is the same as in scenario C. In this case, the standard migration objects will create an installation structure for the active legacy contract starting after the last legacy-billing period. For the period before and for the older closed contract the legacy billing documents is to be migrated as an EBF billing document.

6.3.2.7 Relevant for all Scenarios

The date, where the installation structure has to start, can be defined as follows:

- We recommend that you start the installation structure from the date of the last real meter reading. Please note that a real meter reading result is required in case of DPC. The real meter reading has been used for billing in the legacy system and will be migrated to the SAP system with the meter reading reason *Period Billing* or as an *Interim Meter Reading with Billing*. This is because fewer checks on the billing document will be performed during migration.
- The migration of installation structure is necessary (starting from the move-in date) if an active contract is migrated with a migration date before the first billing.
- It is sufficient to start an installation structure history with the migration date if only an inactive contract is to be migrated.
- With an existing installation structure in the SAP system, there is the choice migrating all the billing documents using the BILLDOC or BILLDOCEBF migration objects. In the first

case, a correction takes place with the `EABICO` standard correction transaction in SAP for Utilities. In the latter case, the correction is done with the `EBF`.

- Migrate contracts independently of the migration object with which the billing documents are migrated. This is valid also in case of historical (inactive) contracts.

6.3.2.8 Further Information about EBF

Further documentation with many implementation details is available in the Cookbook Easy-Bill Correction Framework, which you can find in the **SAP Service Marketplace** at <http://service.sap.com/utilities>.

6.4 Historical Rate Data

The migration of historical rate data for an installation is executed in several steps, to ensure that missing rate data does not result in error messages during the data migration.

1. Configuration in the Implementation Guide (IMG)

Change the configuration so that the specification of a rate type is not required for the installation of devices and registers for billing purposes (`RATETDNES` and `RATETRNES` system parameters). Figure 6-7 shows the configuration path.

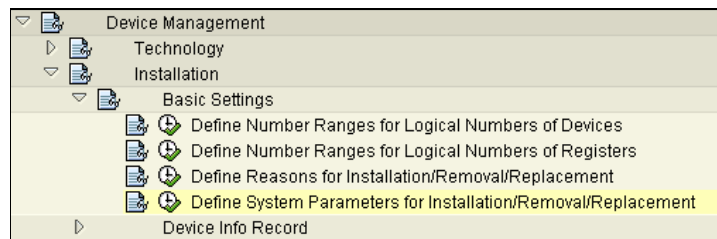


Figure 6-7: IMG Configuration Path

2. Creation of the installation (INSTLN migration object)

Use the `INSTLN` migration object to create an installation with the data relevant for billing, such as the billing class, meter reading unit, and rate category.

3. Change of the installation (INSTLNCHA migration object)

Use the `INSTLNCHA` migration object to create or change the billing relevant data in an installation (for more information, see chapter 7.11.2 *Migration of multiple time slices for an installation*).

Change of the device history (`INST_MGMT` migration object)

Use the `INST_MGMT` migration object to create the device history of the installation.

Creation of additional meter reading results (`METERREAD` migration object)

Use the `METERREAD` migration object to migrate the meter reading results that are required to make billing-relevant changes to rate data.

4. Creation of the rate data for the devices installed in the installation

Use the `DEVICERATE` migration object to create or change the rate data for the device history of the installation.

6.5 Historical Consumption

There are various aspects and functions of the consumption history that affect the data transfer from the legacy system to the SAP system. Data for the consumption history is required for the following reasons:

○ Information purposes

Use the `FACTS` migration object to migrate installation facts to certain operands. Then use the `EAMACF` transaction to create a consumption history (`DBERCHV` database table).

- **Graphical consumption information on the bill**
The necessary information can be migrated through installation facts for certain operands.
- **Influence for the next billing**
The necessary information can be migrated through installation facts for certain operands
- **Estimation of meter readings**
In this case, migrate individual period consumptions (CONSUMPT migration object) for the entire future period in which an estimation is not possible due to missing meter readings. Historical meter readings can be used if billing document history is to be migrated.

6.6 Device Group


It is common to group meters in a device group together with transformers. The following is the simplest approach to group, install, and build the necessary device relationships:

- Migration of devices and transformers (DEVICE migration object)
- Grouping of all transformers (DEVGRP migration object)
- Installation of devices, transformers (INST_MGMT migration object)

Figure 6-8 shows the sequence of the data records in the import file as well as the relevant fields of the respective automation structures for an installation of a device with *n* transformers.

Oldkey	Rec.DataType	EQUFRE / ZWNUMMERE	EQUFREU	WANDRE
<oldkey>	INTERFACE			
<oldkey>	AUTO_ZW	<D:device>/001		
<oldkey>	...			
<oldkey>	AUTO_ZW	<D:device>/00n		
<oldkey>	AUTO_GER		<D:transformer 1>	
<oldkey>	
<oldkey>	AUTO_GER		<D:transformer n>	
<oldkey>	AUTO_GER		<D:device>	<transformer 1>
<oldkey>	&ENDE			

Figure 6-8: Sequence of Structures for Grouped Devices and Transformers

 The two relevant fields of the AUTO_GER structure are the EQUFREU (<D:xxxxx>) and WANDRE (<T:xxxxx>) fields. In the last AUTO_GER data record, the equipment number of the device is transferred as well as the equipment number of one of the transformers of the device group of the transformers. In addition, no technical installation is required before grouping the devices, transformers, and (ARCR).

6.7 Disconnection and Reconnection

You must differentiate for the migration of disconnections between different scenarios. The scenarios differ in the status of the disconnection process in the legacy system, and in the customer-specific implementation of the disconnection process in SAP (for example, the use of workflows or service orders). Several migration objects are used depending of the scenario.

The migration of disconnection documents, disconnection orders, and disconnection entries, as well as, the transfer of reconnections from the legacy system into the SAP system, takes place in several steps. You can only transfer disconnections after installations (INSTLN migration object) and devices (DEVICE migration object) along with the installation of devices in installations that have been successfully migrated. Depending on the scenario,

you must then successfully migrate contracts (MOVE_IN migration object) and dunned open items (DOCUMENT and DUNNING migration objects).

The following restrictions apply for the possible disconnection reasons for the disconnection document:

- The contract account must be transferred in the ACOCUNT field for a disconnection reason *Disconnection Dunning Level Reached*
- The installation must be transferred in the ANLAGE field for the disconnection reason *Vacant Status Disconnection*

Usually, all pending service orders for disconnection and reconnection are closed in the legacy system before the go-live. This is the preferred scenario as it avoids the creation of pending service orders in the SAP system. In some cases, a creation of a disconnection workflow is required. The customer might decide to have the customer calling the call center and request a reconnection. This avoids the creation of workflows during data migration. After the presentation of the possible scenario configuration details are presented to prepare the SAP system for the creation of workflows during data migration.

Disconnection/Re

- **Disconnection (dunning level) - disconnection order not yet completed**
 1. Creation of open items (DOCUMENT migration object). Do not transfer a value in the CHECK_DOC_OFF field to allow a check of the existence of dunned open items.
 2. Creation of dunning history (DUNNING migration object)
 3. Creation of disconnection document (DISC_DOC migration object)
 4. Creation of disconnection order (DISC_ORDER migration object). You must populate the ORDERWERK and ORDERCODE fields in order to create a service order or a service notification.
 5. Start of disconnection workflow (DISC_EVENT or DISC_WF migration object)
- **Disconnection (dunning level) – disconnection order completed**
 1. Creation of open items (DOCUMENT migration object). Do not transfer a value in the CHECK_DOC_OFF field to allow a check of the existence of dunned open items.
 2. Creation of dunning history (DUNNING migration object)
 3. Creation of disconnection document (DISC_DOC migration object)
 4. Creation of disconnection order (DISC_ORDER migration object). You must populate the ORDERWERK and ORDERCODE fields in order to create a service order or a service notification.
 5. Creation of disconnection entry (DISC_ENTER migration object)
 6. Start of disconnection workflow (DISC_EVENT or DISC_WF migration object)
- **Reconnection (dunning level) - reconnection order not yet completed**
 1. Creation of disconnection document (DISC_DOC Migration object)

The open items that have led to a disconnection in the legacy system have already been cleared (the customer paid). You must transfer the value 'X' in the CHECK_DOC_OFF field to suppress a check of the existence of dunned open items.

Creation of disconnection order (DISC_ORDER migration object). You must populate the ORDERWERK and ORDERCODE fields in order to create a service order or a service notification.
 2. Creation of disconnection entry (DISC_ENTER migration object)
 3. Creation of reconnection order (DISC_RCORD migration object). You must populate the ORDERWERK and ORDERCODE fields in order to create a service order or a service notification.
 4. Start of disconnection workflow (DISC_EVENT or DISC_WF migration object)

- **Reconnection (dunning level) - reconnection order completed**
 - 5. Creation of disconnection document (DISC_DOC Migration object)
The open items that have led to a disconnection in the legacy system have already been cleared (the customer paid). You must transfer the value 'X' in the CHECK_DOC_OFF field to suppress a check of the existence of dunned open items.
Creation of disconnection order (DISC_ORDER migration object). You must populate the ORDERWERK and ORDERCODE fields in order to create a service order or a service notification.
 - 6. Creation of disconnection entry (DISC_ENTER migration object)
 - 7. Creation of reconnection order (DISC_RCORD migration object). You must populate the ORDERWERK and ORDERCODE fields in order to create a service order or a service notification.
 - 8. Create reconnection (DISC_RCEN migration object)
 - 9. Close disconnection document (DISC_CLOSE migration object)
- **Disconnection (on request/vacancy) - disconnection order not yet completed**
 - 1. Creation of disconnection document (DISC_DOC Migration object)
Create disconnection order (DISC_ORDER Migration object). You must populate the ORDERWERK and ORDERCODE fields in order to create a service order or a service notification.
- **Disconnection (on request/vacancy) - disconnection order completed**
 - 1. Creation of disconnection document (DISC_DOC Migration object)
Create disconnection order (DISC_ORDER Migration object). You must populate the ORDERWERK and ORDERCODE fields in order to create a service order or a service notification.
 - 2. Creation of disconnection entry (DISC_ENTER migration object)
- **Disconnection (on request/vacancy) - reconnection order not yet completed**
 - 1. Creation of disconnection document (DISC_DOC Migration object)
 - 2. Create disconnection order (DISC_ORDER Migration object). You must populate the ORDERWERK and ORDERCODE fields in order to create a service order or a service notification.
 - 3. Creation of disconnection entry (DISC_ENTER migration object)
Creation of reconnection order (DISC_RCORD migration object). You must populate the ORDERWERK and ORDERCODE fields in order to create a service order or a service notification.
- **Reconnection - reconnection order completed**
 - 1. Create reconnection order (DISC_RCORD migration object)
 - 2. Create reconnection (DISC_RCEN migration object)
 - 3. Close disconnection document (DISC_CLOSE migration object)

The SAP system must be prepared in case disconnection workflows have to be created during the go-live. This is because the number of started workflows in a short time period is exceptionally high and cannot be compared with the execution of the dunning activity run (transaction FFPVB) as part of the daily batch chain.

The system resources required for processing a high volume of workflows (such as dialog work processes) are exhausted very quickly and the system is easily overloaded. You must note that the SAP disconnection workflow and the processing of workflow events at the start of the SAP disconnection workflow must always take place as the last step of the migration of disconnections.

For a successful creation of workflows the event linkage (transaction `SWETYPV`) for the `DISCONNECT` object category, `CREATED` event, and `WS20500080` consumption category (or alternatively for the customer-specific consumption category) must be activated.

Select the **Enable Event Queue** field to use the event queue to process events. In this case, you must also check the settings for the event queue (transaction `SWEQADM`). The event queue allows the delivery of workflow events in a controlled way.

6.8 Sample Lots

Execute the migration of sample lots, their devices, and the corresponding information from the legacy system in the following way:

1. Create sample lots (LOT migration object)
A lot is created with the year of its first sample
2. Create devices (DEVICE migration object)
When a device belongs to a lot, it has also information allocated about its check status within the lot. You can transfer this information when you create a device.
3. Complete work on sample lots (LOTFINAL migration object)
The lot is determined by the lot devices and lot information stored in the devices. It updates the counters and drawing date for an analysis, determination, and drawing of the lot devices.

6.9 Point of Delivery

The migration of points of delivery (POD) and services during the migration can take place in different procedures, depending on the specific needs of the project.

- **Procedure in Case of Non-Deregulated Installations**
A (deregulation) point of delivery without a point of delivery ID is created when an installation is created (INSTLN migration object). For a deregulated market no further actions are required, so you do not need to use the `POD` structure.
- **Procedure in Case of Deregulated Installations**
 1. Create the installation (INSTLN migration object)
A (deregulation) point of delivery is created when an installation is created. In the process, you can use the `POD` structure to control how the corresponding point of delivery is created.
 - If the `POD` structure is migrated containing a category and an external PoD number, a point of delivery is created with this external number and is flagged as a deregulation point of delivery.
 - If the `POD` structure is migrated containing a category and no external PoD number is migrated, the point of delivery is given an internally assigned number.
 2. Create the point of delivery services (PODSERVICE migration object)
You can use the `EXT_UI` field of the `ESERVICED` structure to transfer the name of the PoD for which the service will be created.
 3. Create additional points of delivery (POD migration object)
Use the `POD` migration object to create additional PoDs (such as technical points of delivery).
 4. Change the data of existing points of delivery (PODCHANGE migration object)
You can change an existing point of delivery (for example, by assigning an additional technical point of delivery to a device).

○ Additional Special Features in the Case of EDM (Energy Data Management)

Use the POD migration object to create additional PoDs, such as, technical points of delivery. You can also assign devices or registers to the PoD by using the TECH_ANLAGE, TECH_LNR, or TECH_LZW structures.

6.10 Service Provider

Execute the migration of service provider relevant data in the following way:

1. Generate BAPI migration objects using the following function modules as service function modules of the new migration objects
 - ISU_MIG_SP_MASTERDATA Header data and historical data
 - ISU_MIG_SPAGREE_PARA Agreements and parameter configurations
 - ISU_MIG_DEXDEFSP Definition of the data exchange

For more information how to generate BAPI migration objects, see chapter 3.4.2 *Generation of a BAPI Migration Object*.

2. Create the service provider master data with the generated migration object for function module ISU_MIG_SP_MASTERDATA.
3. Create the service provider agreements and parameter configurations with the generated migration object for function module ISU_MIG_SPAGREE_PARA.
4. Create the data exchange definition with the generated migration object with the generated migration object for function module ISU_MIG_DEXDEFSP.

6.11 Energy Data Management (EDM)

Figure 6-9 shows the data model for EDM. For more information, see the EDM Cookbook. On the homepage of SAP for Utilities, choose *Media Center* → *Energy Data Management* → *Literature*.

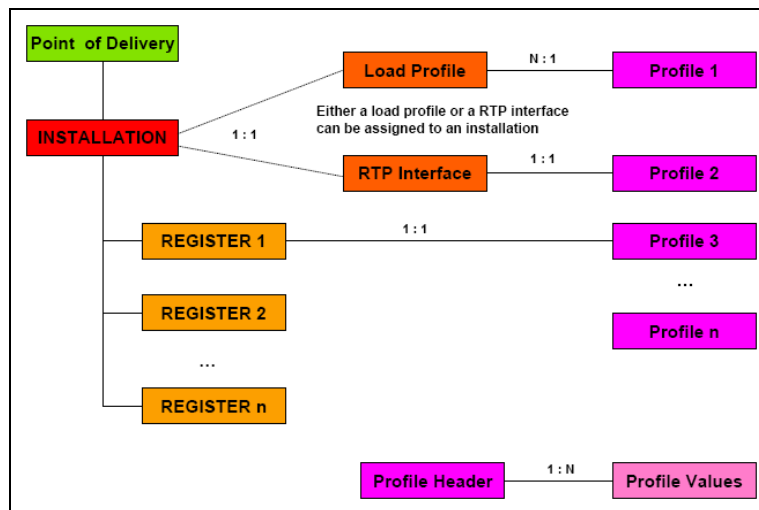


Figure 6-9: EDM Data Model

Execute the migration of the EDM relevant configuration in the following way:

1. Create installations (INSTLN migration object)
2. Create device installations (INST_MGMT migration object)
3. Create profile header (PROFHEAD migration object)

You can create profile headers independently of other migration objects by using the PROFHEAD migration object. You cannot create synthetic profiles and formula

profiles using this migration object. For more information how to migrate formula profiles, see 7.11.16 *How to migrate formula profiles?*

4. Create installation specific RTP interface (RTP_INTERF migration object)
The `ISU_EEDM_S_AUTO_LINK` function module can be used to create and activate the installation specific RTP interfaces (see implementation details below).
5. Create EDM load profiles (LOADPROF migration object)
Load profiles and their allocation to installation can now be migrated.
6. Create and assign a register (PROFASSIGN migration object)
You must define a register code and interval length for the registers in question. To allocate the interval length and the unit of measurement of the register they must correspond to the data entered in the profile header.
7. Create profile values (PROF_VALUES migration object)
You can migrate profile values using the `BAPI_ISUPROFILE_IMPORT` function module, the `IMPORT` method of the `ISU_PROFILE` object type in a custom program, or in a generated BAPI migration object based on the BAPI. For more information about the procedure for the generation of this kind of migration object, see 3.4.2 *Generation of a BAPI Migration Object*.

You cannot use the `ISU_EEDM_S_AUTO_LINK` service function module to generate a BAPI migration object due to its complex interface. To take advantage of a BAPI migration object, you must develop a function module with a simplified interface that calls the `ISU_EEDM_S_AUTO_LINK` function module. Figure 9-1 shows the necessary ABAP code. For more information about the procedure to generate the new `RTP_INTERF` BAPI migration object using the developed `Z_MIGRATION_RTP_INTERFACE` function module, see 3.4.2 *Generation of a BAPI Migration Object*.

6.12 Installment Plan

An installment plan is an agreement in which a business partner consents to pay off an outstanding receivable in partial payments. An installment plan represents a substitute document. The following are alternatives to migrate installment plans:

- Migrating installment plans with only the non-paid installments
 1. Create open item (DOCUMENT migration object)
Create the open item with an amount that corresponds to the sum of the non-paid installment of the installment plan.
 2. Create installment plan (INSTPLAN migration object)
Create the installment plan with the non-paid installments.
- Migrating installment plans as they have been originally created in the legacy system
 1. Create open item (DOCUMENT migration object)
Create the open item with the original amount. Disregard that an installment plan has been created and is already partially paid.
 2. Create installment plan (INSTPLAN migration object)
Create the installment plan as you have created it in the legacy system, with all paid and non-paid installments.
 3. Create payments (PAYMENT migration object)
The already paid installments are cleared by creating payments. The document number that you set the payment to is the same document number of the substitute document (document number of the installment plan).

6.13 Dunning History

The migration of dunning history depends on whether the collection management is implemented based on dunning levels and corresponding dunning procedures or with the collection strategy.

- Migrating dunning history based on dunning levels and dunning procedures
 1. Create open item (DOCUMENT migration object)
 2. Create dunning history (DUNNING migration object)
- Migrating dunning history based on collection strategies
 1. Create open item (DOCUMENT migration object)
 2. Create collection strategy

Create a new BAPI migration object with the `FKK_CM_DUNNING_DATA_CREATE` service function module. For more information about the procedure for generating a BAPI migration object, see 3.4.2 *Generation of a BAPI Migration Object*.

The `STEP` (collection step of dunning) and `AUSDT` (date of issue) fields of the `IS_COLLDUNNHEADER` structure are mandatory. You can fill the `GRPFIELD` (Grouping Field for Dunning) optional field in the FI-CA event 0312 if you are creating several dunning groups for each master data group. The `IT_ITEMS` internal table contains the open items of the dunning group, whereas, the `IT_REDUCTIONS` internal table contains the dunning reductions. You can fill these in the FI-CA event 0335 if you have to consider reductions due to dispute cases.

6.14 Collection Agency

A collection agency is a company that collects payment on unpaid bills. For this, debts in the SAP system are released to one or several private collection agencies (PCA) to negotiate payment terms with these customers (for example, in agreeing on an installment plan).

The SAP standard is not fully prepared yet it supports the migration of data relevant to the PCA with all the details required, such as:

- Date for submission of receivables to PCA as it is in the legacy system
- Customer fields created in the `CI_FKKCOLL` customer include

This will require either a temporary modification in the SAP standard modules or changes in a copy of the standard modules. The latter option might be better because it is a modification-free approach and the copied module can be deleted after the go-live. You must consider that the effort to implement the strategy must be as little as possible.

The migration of data related to private collection agencies is executed in the following way:

1. Create open items either fully or partially released to the PCA (DOCUMENT migration object). The `XBLNR` field (external document number) is populated with the document number that is known by the PCA. This field is also be used in production to interface with the PCA.
2. Create dunning history (DUNNING migration object)
The dunning level reflects the dunning level as it is in the legacy system.
3. Release open items to PCA

The release of the open items for collection is the first PCA specific step. No standard migration object exists to support the migration of this step. SAP offers these options in the standard to release open items to the PCA:

- Transaction `FP03`
- Transaction `FP03M` (mass activity)

You can also generate a BAPI migration object using the `FKK_RELEASE_FOR_COLLECT_AGENCY` standard function module to create rows in

the DFKKCOLL and DFKKCOLLH database tables and update the INKPS field (mark open item as a collection item) in DFKKOP database table.

4. Submit open items to PCA for collection

The following transactions are available to submit released open items to the PCA:

- Transaction FP03D (RFKKCOL2 report)
- Transaction FP03DM (mass activity)

The SAP standard sets the submission date to the system date (SY-DATLO field). This is not recommended as the submission date should be the same as in the legacy system. You can solve this problem by temporarily saving the correct submission date in a custom-table in step three when you are using the generated migration object. You must develop a new event in the migration object. The fields of the custom-table should be MANDT (client in SAP system), OLDKEY (legacy system date), and AGDAT (submission date in legacy system).

We recommend copying the RFKKCOL2 standard report and changing the line indicated in Figure 6-10. Instead of using the SY-DATLO system field, you retrieve the submission date from the custom-table created in step three. After having submitted the open items to the PCA the status of the open items in DFKCOLL database table changes from DFKKCOLL-AGSTA = 01 to DFKKCOLL-AGSTA = 02.

```
* ----- Sort in output sequence by collection agency -----*
SORT t_fkkcoll BY inkpg ASCENDING.
* ----- Write the output list -----*
LOOP AT t_fkkcoll.
  READ TABLE t_fkkop WITH KEY opbel = t_fkkcoll-opbel
                                inkps = t_fkkcoll-inkps.

  IF sy-subrc NE 0.
    DELETE t_fkkcoll.
    CONTINUE.
  ENDIF.

  t_fkkcoll-agdat = sy-datlo. " ← modification of submission date
  t_fkkcoll-agsta = const_agsta_abgegeben.

  CLEAR: t_fkkcoll-rudat, t_fkkcoll-rugrd.

  IF NOT t_fkkcoll-betrz IS INITIAL OR
     NOT t_fkkcoll-ninkb IS INITIAL.
    t_fkkcoll-betrw = t_fkkcoll-betrw - t_fkkcoll-betrz -
                      t_fkkcoll-ninkb.
    CLEAR: t_fkkcoll-betrz, t_fkkcoll-ninkb, t_fkkcoll-nrzas.
  ENDIF.

  MODIFY t_fkkcoll.
```

Figure 6-10 Modification of the RFKKCOL2 Report

5. Recall released open items

The recall of the open items from the PCA for collection is the last PCA specific migration step. No standard migration object exists to support the migration of this step. We offer the standard option to recall open items from the PCA with transaction FP03U (RFKKCOLR report). With this transaction, all submitted open items are recalled, as only migrated ones exist in the SAP system.

The SAP standard sets the recall date to the system date (SY-DATLO field). This is not recommended as the recall date should be the same as in the legacy system. To solve this problem, temporarily save the correct recall date in a custom-table, as shown in step three, when you use the generated migration object (this adds the RUDAT field to the created custom table).

We recommend that you copy the RFKKCOLR report and change the line indicated in Figure 6-11. Instead of using the SY-DATLO system field, the submission date is retrieved from the custom-table created in step three.

```

LOOP AT t_recall_coll.

  READ TABLE t_reassign_coll WITH KEY opbel = t_recall_coll-opbel
                                     inkps = t_recall_coll-inkps.

  IF sy-subrc = 0.
    t_reassign_coll-agsta = const_agsta_frei gegeben.
    t_reassign_coll-agdat = '00000000'.
    t_reassign_coll-xblnr = t_recall_coll-inkgp.
    MODIFY t_reassign_coll INDEX sy-tabix.
  ENDIF.

  CLEAR h_inkps.

  IF t_recall_coll-agsta EQ const_agsta_abgegeben OR
     t_recall_coll-agsta EQ const_agsta_erfolgrlos OR
     t_recall_coll-agsta EQ const_agsta_sub_erfolgrlos OR
     t_recall_coll-agsta EQ const_agsta_recall OR
     t_recall_coll-agsta EQ const_agsta_rec_erfolgrlos OR
     t_recall_coll-agsta EQ const_agsta_teilbezahlt OR
     t_recall_coll-agsta EQ const_agsta_cust_p_pay OR
     t_recall_coll-agsta EQ const_agsta_p_paid.

    t_recall_coll-agsta = const_agsta_recall.
    t_recall_coll-rudat = sy-datlo. " ← modification of submission date
* get recalling reason from posting area 1058
    PERFORM read_buber_1058 CHANGING h_rugrd t_fkkop-applik.
    IF NOT h_rugrd IS INITIAL.
      t_recall_coll-rugrd = h_rugrd.
    ENDIF.
  ENDIF.
  MODIFY t_recall_coll.
ENDLOOP.

```

Figure 6-11 Modification of the RFKKCOLR Report

You can also generate a BAPI migration object using the SAP FKK_RECALL_FROM_COLLECT_AGENCY standard function module.

The recall of the related open items from the PCA is reflected in the status of the open items in DFKCOLL database table from DFKKCOLL-AGSTA = 02 to DFKKCOLL-AGSTA = 09.

6. Create installment plans (INSTPLAN migration object)
7. Reconcile migrated data with PCA

You must not implement a reconciliation process of data submitted to the PCAs only between the legacy system and the SAP system. The reconciliation process must also include the data available at the PCAs. During submission and recollection of the open items, data files can be created in the SAP standard (using the mass transaction FPCI (*Information for Collection Agency*), you can create the data files for the recall in a separate step). These data files implement the interface to the PCA during production operation. You can also use these data files during migration to create information for reconciliation to be sent to the PCA.

6.15 Cash Security Deposit

Execute the migration of security deposits in the following way:

1. Create cash security deposit (SECURITY migration object)
You can classify a security deposit as a non-cash security deposit using the `NON_CASH` field. Otherwise, the system posts a request. You can determine the number of the document using the number of the security deposit in the `OPBEL` field of the `FKK_SEC_REQ` database table.
2. Create payment (PAYMENT migration object)
The payment clears the request generated by creating a cash security deposit. To refer to this request, you create a rule for determining the document number from the security deposit.
3. Create interest information (INTCASHDEP migration object)
This object generates the necessary interest document for credit originating from cash security deposits.

6.16 Budget-Billing Plan

When you migrate budget-billing plans, differentiate between the different types of budget-billing plan procedures (statistical, debit entry, and payment plan procedures).

Budget-billing plans in their pre-migration form are migrated in several steps, which apply to all budget-billing types. The master data must always have the following attributes:

- The `KZABSVER` field (BB procedure) of the `VKP` structure in the contract account must have an appropriate entry (ACCOUNT migration object).
- For the payment plan procedures you must enter the start and finish month, as well as, the payment plan category (`PYPLT` field) on the contract level (in `MOVE_IN` migration object).

Structure a payment or budget-billing plan as follows:

- The same number of line items is transferred for each due date.
- The same number of lines is transferred for each due date and each contract.
- The same number of line items is transferred for a specific main-transaction or sub-transaction for each due date and contract.

There are special requirements for each budget billing procedure as follows:

- **Statistical Procedure**

1. Create the budget-billing plan (BBP_MULT Migration object)
All due dates, even those already paid in the legacy system, are created with the original amount to be paid. Here, the `BETRW` and `BETRO` fields (ordered or open amount) of the `T_EABPS` structure are set to the same value.
2. Create the payment (PAYMENT Migration object)
All the due dates of the generated budget-billing plan that are already (partially) paid are cleared with the correct amount.

- **Debit Entry Procedure**

1. Create the budget-billing plan (BBP_MULT migration object)
All due dates, even those already paid in the legacy system, are created with the original amount to be paid. The following applies:
 - The `BETRW` and `BETRO` fields of `T_EABPS` structure are set to the same value for all due dates for which a debit position has not been run in the legacy system.

- The `BETRW` field is set to the due date amount and the `BETRO` field to zero for all due dates for which a debit position has been run in the legacy system.
- 2. Create unpaid partial bills (DOCUMENT migration object)
You must migrate the partial bills as open items. You can do this in the following ways:
 - Migrate only the partial bills that have a balance not equal to zero in the legacy system.
 - Migrate all partial bills originating from the budget-billing plan. Then migrate the payments of any partial bills that were already paid in the legacy system using the PAYMENT migration object.
- **Payment Plan Procedure**
 1. Create the payment plan (BBP_EXT migration object)
All the due dates still outstanding after the migration date (system date) are structured to be the original amount.
 2. Create open items for the balance forward (DOCUMENT migration object)
The last difference amount determined in the billing or invoicing component of the legacy system is created as an open item for main transaction 0150.
 3. History of the requested amounts
You can save the billing amount due for each contract participating in the payment plan procedure in the `TEPYPM` database table. This applies to previous periods. During invoicing, you can use FI-CA event R950 to enhance any missing history data by using entries in this table. For more information, see the `ISU_CALC_PAYPLAN_AMOUNT` function module documentation.
- **Special BBP_CB_TAB**
If it is necessary to migrate budget-billing plans that are allocated to a collective bill, you must execute the following steps (these actions depend on the budget-billing procedure):
 - **Statistical budget-billing procedure**
 1. Create the budget-billing plans (BBP_MULT migration object)
 2. Create the collective document numbers (BBP_CB_TAB migration object)
 3. Request the budget-billing amounts (transaction `EA12`)
 4. Create the collective bill (transaction `EA10_COLL`)
 5. Create the budget-billing payments (PAYMENT migration object)
 - **Partial bill procedure**
 1. Create the budget-billing plans (BBP_MULT migration object)
 2. Create the collective bill numbers (BBP_CB_TAB migration object)
 3. Create the partial bills (DOCUMENT migration object)
 4. Create the budget-billing payments (PAYMENT migration object)

6.17 Payment Scheme

Execute the migration of payment schemes in the following way:

1. Create open items relevant for the payment scheme (BBP_PAYS migration object)
You must create the open item with the clearing restriction 'R' (items relevant for payment scheme). You can transfer the clearing restriction the `AUGRS` field of the `FKKOP_T` structure.

2. Create payment scheme (BBP_PAYSC migration object)

For more information about payment schemes and the migration of payment schemes, see **Payment Scheme** in the **SAP Service Marketplace** at <http://service.sap.com/utilities>.

3. Create payment (PAYMENT migration object)

The already paid payment scheme relevant open items are cleared by creating payments. The PAYMENT migration object allows the clearing of open item with clearing restriction 'R'.

6.18 Replication to SAP CRM

6.18.1 Data Model

The replication of business objects between SAP for Utilities to SAP CRM has become a part of most SAP for Utilities implementation. Figure 6-12 explains the connection between objects in SAP for Utilities and the corresponding objects in SAP CRM. The integration solution ensures consistency between the objects in both systems (for example, changes to a contract in SAP for Utilities lead to changes to the corresponding contract item in SAP CRM, and vice versa).

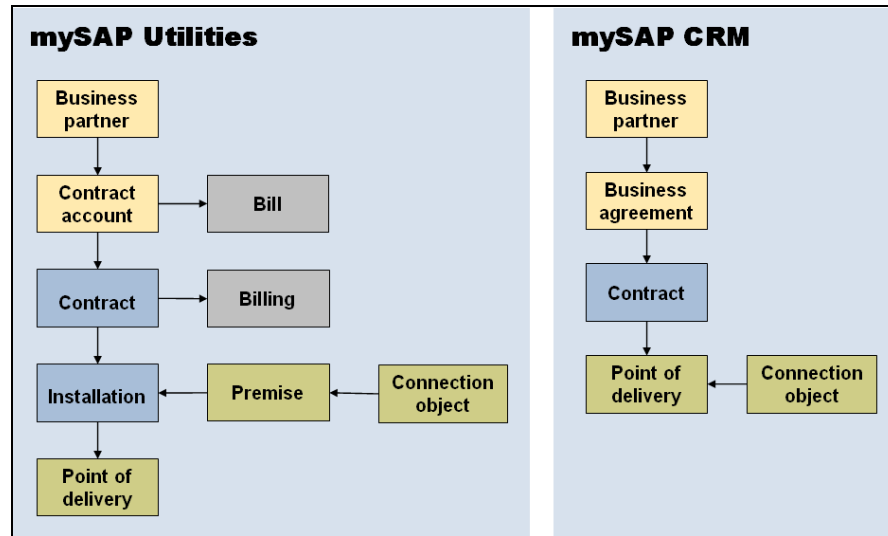


Figure 6-12 Integration Model SAP for Utilities and SAP CRM

6.18.2 Replication Strategies

The main options for a replication strategy together with a go-live and migration strategy are as follows:

- Replication after finishing the data migration process
- Replication during the data migration process

A delta (or online) replication is not a viable option in most cases due to performance problems. The recommended replication method in most cases is the replication with an initial download (for example, request download with transaction R3AR4). With a request download, the initial download can be parallelized per business object. This utilizes the available system resources best.

For more information about documentation and technical details about SAP for Utilities or SAP CRM replication, see **IS-U/CRM-Integration: Replication of Technical Objects** in the **SAP Service Marketplace** at <http://service.sap.com/utilities>.

6.18.2.1 Replication Strategy: Replication After Finishing Data Migration

Defer any replication activities to the end of the data migration process. During data migration, the middleware is deactivated temporarily (for example, with transaction BF11 in the TBE11 database table). Upon completion of the data migration, the middleware is reactivated and the replication for all required business objects is initiated.

6.18.2.2 Replication Strategy: Replication during Data Migration

This is an advanced replication strategy, which allows you to minimize the migration window. The general idea is to initiate the replication of a business object immediately after finishing the data migration of that business object (for example, business partner) while continuing the data migration with the next business object. This parallelizes the replication and data migration processes. It is important to understand the following:

- This replication strategy requires a stabilized data migration process
- Profound knowledge of the middleware and replication process
- The system landscape must support a parallel migration and replication process (for example, separate hardware for each SAP system or no resource sharing). The data migration into SAP for Utilities, as well as the replication process to the SAP CRM system, will increasingly compete for the available system resources. Avoid this to accomplish the migration, as well as the replication, in the shortest time possible.

Use the CRM_REPL_OFF control parameter of the IS Migration Workbench to allow a data migration with an active middleware. The application checks the setting of this control parameter in conjunction with the Customizing in the IMG (see Figure 6-13 for the IMG path). It bypasses the delta replication if the control parameter is set to X. Instead, the identifications of the created objects are temporarily stored in ECRM_TEMP_OBJ database table where they can be selected during a request load with the filter settings for the ECRM_TEMP_OBJ database table and the XREPLCNTL field set to X.

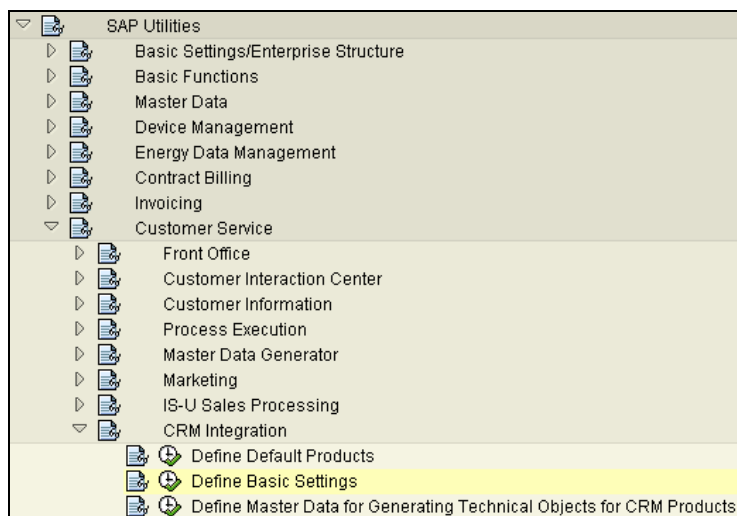


Figure 6-13: IMG Configuration Path

Figure 6-14 shows a control matrix when a delta replication takes place during data migration.

Tech.Obj. Active	CO/PoD Download Active	ISMW CRM_REPL_OFF	Effect
X	X	X	No delta replication – Entry in ECRM_TEMP_OBJ
X	X		Replication
X		(X)	No delta replication Entry in ECRM_TEMP_OBJ
	(X)	(X)	No integration active for technical objects

Figure 6-14: Control Matrix for Delta Replication during Data Migration

7 FAQ (Frequently Asked Questions)

7.1 Motivation

This chapter is a collection of many common problems and questions. Many customer messages have been analyzed in order to present the most common problems and their solutions in this document.

7.2 FAQ Usage of the IS Migration Workbench

7.2.1 One or more fields cannot be found in a migration object

Problem

One or more fields are missing in the field list of a structure, even in the ABAP data dictionary the definition of the underlying structure of the automation data contains the field.

Solution

This problem occurs most after you have added fields to a customer include as part of the enhancement development process. The IS Migration Workbench compares the existing field list automatically during a copy of a migration object. If you have added a field after the copy, the comparison of the field list with the field list of the structure as defined in the data dictionary has to be initiated manually. This can be done for a single structure (double-click the structure to mark it and choose *Auto structure → Compare with DDIC*), or for one or more migration objects at a time, choose *Utilities → More Utilities → Compare → Dictionary: Total comparison*.

7.2.2 Why is it required to transfer default information?

Problem

During data migration much more information has to be transferred in the import file instead of being completed automatically as in the related dialog transaction. Why is the automation data during data migration not completed in the same way in order to reduce the size of the import file?

Solution

During the execution of dialog transactions, the data on the screen is completed by default values or by values relevant for the already entered data. For this, the dialog transaction has to access the database in order to retrieve the necessary information. This is not desirable during data migration because it affects performance. If the data cannot be transferred from the legacy or transformation system, the automation data has to be completed during runtime in the load report in implementing own coding to retrieve the data from the database (for more information about how to implement own coding in a load report, see chapter 4.3 *Custom Code in a Load Program*).

7.2.3 Error EM 024: Object in import file header does not correspond

Problem

While trying to start a data import the IS Migration Workbench displays the error EM 024 *Object in import file header does not correspond to the migration object*.

Solution

The first data record in the import file contains information (for example, migration company and migration object) for which migration object the import file was created. The IS Migration Workbench compares this information with the selected migration object and allows a data import for the selected migration object only if the information matches. This is to avoid a data import of a file you have created for a different migration object.

7.2.4 Error EM 044: Change not possible due to blocking status

Problem

While changing to the data import screen the IS Migration Workbench displays the error EM 044 *Change not possible due to blocking status of the migration object*.

Solution

Change the blocking status of the migration to an appropriate status (for more information about the blocking status, see chapter 2.9 *Authorization*)

7.2.5 Warning EM 188: Mandatory indicator deleted

Problem

While changing the field Customizing in the field maintenance screen, the warning EM 188 *Mandatory indicator for <field> deleted; indicator set in original object* is displayed. Can the warning be ignored?

Solution

The warning explains that you have cleared the *Mandatory* indicator (**Req.Fld** field) in the field maintenance screen even though the SAP system delivers this field marked as mandatory. Figure 7-1 shows the message you may have confirmed, but ignored, while continuing the migration Customizing.

You should only clear this field and ignore this warning if you are sure that this field is not required for the processing of the field and the business object to be created or changed. Not transferring a value in this field may lead to unexpected results in the application. Short dumps may occur.

You deleted the required entry field indicator for field BU_RLTYP. This indicator is set in original object PARTNER of company SAP, which is delivered by SAP. The required entry field indicator shows, that this field is a 'required field' and therefore must always be supplied with a value for migration.

Figure 7-1: Warning When Deleting the Mandatory Field Indicator

7.2.6 Error EM 104: Table x not supplemented

7.2.6.1 General Problem

Problem

The application raises the error EM 104 *The table for x (Oldkey) was not supplemented*

Solution

The load program expects the service function module to return consistent status information. In case of a successful update of the database (either the creation of, or an update of, an object) the service function module must return a return code zero in the SY-SUBRC system variable and the DB_UPDATE parameter is set to X. Any other combination is invalid. It indicates that the application did not do any database update based on the data provided. Most likely, the provided data matches the data of the already saved object.



There is an exceptional error handling for BAPI migration objects. A BAPI migration object does neither raise an exception (SY-SUBRC <> 0 system variable) nor uses the DB_UPDATE return parameter. Instead, the information in the RETURN parameter is used to evaluate the success of a database update. The load program assumes a successful migration if the service function module returns no error message in the RETURN parameter.

7.2.6.2 Error EM 104 During Migration of DEVICEMOD

Problem

During migration of a device modification, the application raises the error EM 104 *The IS-U table for <oldkey> was not supplemented.*

Solution

The most likely cause for this error is that the change indicator is not set. The change indicator is in the DEV_FLAG, REG_FLAG and REG_MOD_FLAG structures. In these structures, a field (field length one character) with the same name can be found in the corresponding DEV, REG and REG_MOD structures. To change a device parameter the respective change indicator in the *_FLAG structure, set to X. This is because a space might also be a valid parameter value.



If the transformation ratio of a device has to be changed (DEV-UEBERVER field), the value 'X' has to be transferred in the DEV_FLAG-UEBERVER change indicator.

7.2.6.3 Error EM 104 during Migration of PODSERVICE

Problem

During migration of a service of a PoD (Point of Delivery) the application raises the error EM 104 *The IS-U table for <oldkey> was not supplemented.*

Solution

The most likely cause for this error is that you have created the PoD service during the migration of the contract (MOVE_IN migration object). It is possible to suppress the processing of POD services during the creation of contracts. If the MOI_POD_SERV_CH_OFF migration control parameter is not used or set to space, the application takes the default values for non-billable point of delivery services into account, and creates them for the related installation as configured in the IMG.

7.2.7 How can TEMKSV database table become part of a client copy?

Problem

The TEMKSV database table is not taken into account during a client copy. Why is this, and how can this database table be included in a client copy?

Solution

In the TEMKSV database table for all migrated objects a row can be found (for more information about the key and status management, see chapter 2.5.2 *Implementation*). The TEMKSV database table is defined with the delivery class L (table for storing temporary data that is delivered empty). Because of this delivery class, this table is excluded from a client copy. If the database table should be part of a client copy, modify the table definition in the data dictionary by changing the delivery class from L to a transportable class, for example, A, and incorporate the TEMKSV database table in the client copy profile.

7.2.8 Can the rows in TEMKSV database be deleted after go-live?

Problem

A very high number of rows are stored in the TEMKSV database table. Can these rows be deleted after go-live in order to save space in the database?

Solution

For every successfully migrated data object, the relationship between the legacy system key and the SAP system key is stored in the TEMKSV database table (for more information, see chapter 2.5 *Key and Status Management*). The IS Migration Workbench does not require this information anymore after the successful completion of the migration process. From a data

migration prospective, you can delete all rows in the `TEMKSV` database table after the go-live. However, you cannot delete the rows containing information that is still required by business processes in the SAP system, such as a manual or automatic verification and correction processes of migrated data or interfaces to external systems, which still use the legacy system key as an identification of migrated business objects.



Often, the relationships for migrated business partners, contract accounts and contracts are still in use after the go-live. We recommend to archive the content of the `TEMKSV` database on a CD or DVD and subsequently delete all rows in the database table that are not required by an active business process. If necessary, you can still restore the content of the `TEMKSV` database table from CD at any time. You may want to implement a deletion plan in two steps:

- Within 3 months after go-live: Deletion of the majority of the rows
- Within 12 month after go-live: Deletion of the rows no business processes uses



There is no archiving object available for the `TEMKSV` database table. You must delete rows in this database table with either an own ABAP program or database commands on database command level.

7.2.9 Why to re-generate the load report after a transport request?

Problem

It is required to regenerate the load reports for the migration objects when a transport request was imported containing these migration objects. Sometimes the regeneration for all migration objects is required, even when the transport request does not contain any migration object. Why is regeneration of the load report required?

Solution

Migration objects are transported with the `TEMO` logical transport object (transaction `SOBJ`). Apart from the piece list of the `TEMO` logical transport object (references to the relevant rows in various database tables), the transport object is defined by an after import method (`ISU_M_TEMO_AFTER_IMP` function module). It changes the status of the related migration object to force a regeneration of the load report.

The regeneration is required because an update of the present migration Customizing is expected in the transport request (why else the migration object is included in the transport request?). It has frequently occurred that a data import has been executed with a load report that was generated based on the migration Customizing before the Customizing was updated with the transport request. This led to unexpected errors during data migration. Therefore, a regeneration of the load report is obligatory after an import of a transport request that contains a migration object.

The regeneration of the load program is also required when the transport request contains an updated version of one of the modules of the load program generator. The regeneration of the load program makes sure that the ABAP code in the load program is generated with the most recent version of the load program generator.

7.2.10 Data Migration works in one system but not in a second system

Problem

Data migration works in one system environment correctly but not in the second environment. The SAP system configuration and the Customizing is the same as in the first environment, also the import files are the same. However, during data migration data import jobs cancel only in the second system environment. The systems use a UNICODE code page. What might be the root of the problem?

Solution

Among many reasons, the binary data, you transferred in the import file, might have caused the problem. In most cases, you transfer only character information in the import file. However, you can configure field rules of the fields, which are in the ABAP data dictionary defined with a binary data type (for example, integer or packed data), to be transferred in the SAP internal binary format instead of transferring them in the character format. The load program moves binary information from the data record of the import file to the respective field in the automation data structure by a byte-move.

In dependency of the hardware (and sometimes of the operating system), the binary representation of multi-bytes binary data types in the main memory of the system may differ. In big-endian architectures, the bytes on the left (those with a lower address) are most significant. In little-endian architectures, the bytes on the right (those with a higher address) are most significant. This means, that the process to create import files (for example, extraction or transformation process) for binary information depends on the hardware architecture. In order to implement a process, that is independent from the used hardware architecture, use always the character representation (**Screen Display**) instead of the **Internal format** for the data migration process. Figure 7-2 shows two examples with the recommended Customizing on the sub screen for the *Transfer* field rule.

Integer data type		Currency data type	
<input checked="" type="radio"/> Screen display		<input checked="" type="radio"/> Screen display	
Data Type	INT4	Data Type	CURR
No. of Characters	10	No. of Characters	13
Decimal Places	0	Decimal Places	2
Output Length	10	Output Length	18
<input type="radio"/> Internal format		<input type="radio"/> Internal format	
ABAP type	I	ABAP type	P
Internal Length	4	Internal Length	7

Figure 7-2: Customizing of Fields with Binary Data Type

7.2.11 Why does delta replication not work during data migration?

Problem

When the respective business object is created with the dialog transaction, it is replicated successfully thus showing, that the middleware is configured correctly. Why does delta replication to CRM (for example, business partner) not work during data migration with the IS Migration Workbench?

Solution

The CRM_REPL_OFF migration control parameter in the IS Migration Workbench controls whether a delta replication to CRM takes place during data migration. The replication is disabled by default. If an online (delta) replication of certain business objects is required, the CRM_REPL_OFF control parameter must be set to space.



We do not recommend that you replicate business objects through the delta queue. Instead, use an initial download (for example, request download with transaction R3AR4). For more information, see **IS-U/CRM-Integration: Replication of Technical Objects** in the **SAP Service Marketplace** at <http://service.sap.com/utilities>.

7.2.12 Why does the CRM_REPL_OFF control parameter not work?

Problem

Business Documents (BDocs) are created online during the migration of equipments and can be monitored with transaction `SMW01`. Why does the `CRM_REPL_OFF` migration control parameter not work?

Solution

The `CRM_REPL_OFF` migration control parameter in the IS Migration Workbench controls the replication of the relevant objects of SAP for Utilities or SAP CRM integration (business partner, contract account or business agreement, IBase and contract). A replication of objects (for example, equipments) due to the activation of further solutions, such as, SAP Mobile Asset Management (MAM), cannot be controlled by this control parameter. Instead, temporarily disable additional applications during data migration (transaction `BF11`).

7.3 FAQ Migration of Addresses

7.3.1 Can the address validation be deactivated during data migration?

Problem

During data migration, many business objects (such as, business partners), are rejected due to incorrect addresses. Subsequently, even more business objects fail to migrate due to the missing superior business object. Should the address validation against the regional structure be deactivated in such a situation? Is this an option during the go-live? How can the address validation be deactivated?

Solution

There is an option available to activate and deactivate the address validation against the regional structure. Figure 7-3 shows the configuration path in the IMG.

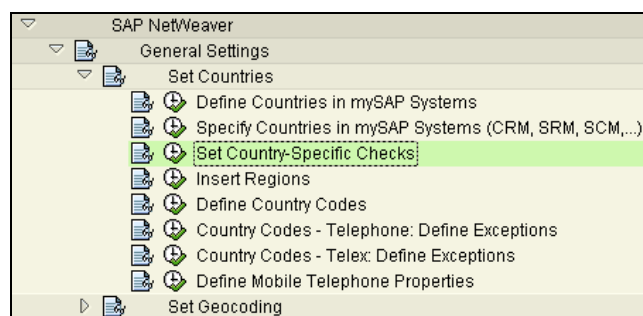


Figure 7-3: IMG Configuration Path

After the selection of the country, the maintenance screen is displayed as shown in Figure 7-4. The **City file active** indicator controls the validation of addresses.

The image shows the 'Maintenance Screen for Country Specific Checks' for country DE. The screen is divided into several sections:

- Country Key:** DE
- Key for the bank directory:** Bank Key 1, Bank number
- Formal checks:** A table with columns for Length and Checking rule.

	Length	Checking rule
Postal code length	5	4 Length to be kept to exactly, numerical, with
Bank account number	10	2 Maximum value length, numerical, without gaps
Bank number length	8	4 Length to be kept to exactly, numerical, with
Post bank acct no.	10	1 Maximum value length, without gaps
Tax Number 1		
Tax Number 2		
VAT registration no.	20	5 Maximum value length
Length of Bank Key		
- Further checks:**
 - ☒ Bank data
 - ☐ Other data
 - ☐ Postal code req. entry
 - ☐ P.O.box code req. entry
 - ☒ City file active
 - ☐ Street postcode

Figure 7-4: Maintenance Screen for Country Specific Checks

We recommend keeping the address validation (regional structure check) active during data migration. This is valid for all objects with an address allocated (for example, business partner). Therefore, it is not a viable option during the go-live to

- migrate addresses when the regional structure validation is deactivated and to activate the validation after the go-live.
- migrate addresses in a first try when the regional structure validation is activated, and in a second run when the validation is deactivated to allow business objects to be migrated even with invalid addresses.

Invalid addresses may lead to a major impact on the business after the go-live. The major consequences could be as follows:

- You cannot save any change of a business partner if the allocated address is still invalid. This affects almost any business process that involves a business partner. This causes problems for the call center agents as well as any back-office operation. Specifically, the impact on the call center operations must not be underestimated. For example, it can happen that a calling customer is asking for a service but the service cannot be saved because the caller cannot provide a correct address.
- Any evaluation of data in BI (business intelligence) based on regional structure criteria (like street or city) is not possible. If you have only partially validated the saved addresses, the results of an evaluation can lead to inaccurate results.
- Later attempts for address cleansing will cost much more than to cleanse addresses before the go-live. Take into consideration that recently created data is distributed between systems (for example, between an ERP system, CRM system, and BI system).

You can identify invalid addresses and addresses that have been created when the regional structure check is deactivated, in the `ADRC` database table where the `CITY_CODE` and `STREET_CODE` fields are saved without a link to the respective city or street in the regional structure.

7.3.2 How to deal with uncleansed addresses in test migration cycles

Problem

Understanding that the cleansing of addresses is a major and lasting process of nearly all data cleansing projects, a temporary solution must be implemented to deal with uncleansed addresses until the go-live.

Solution

It is not an option to change the Customizing in the IMG during data migration. This requires that either the SAP system is opened for Customizing changes or transport requests have to be imported during data migration to change the respective Customizing parameter. This is not recommended. Instead, use the `ADDR_ACCEPT_REG_DATA_ERROR` function module to control the validation of addresses. This function module can be implemented in an event at the `CALL02` generation point and can be activated and deactivated when necessary. For more information about how to add an event in the load program, see 4.3.3 *Code on Report*. The `ADDR_ACCEPT_REG_DATA_ERROR` function module must be called with `ACCEPT_ERROR` parameter set to `X`.

Initially, the event should be deactivated to force the application to reject all business objects with an invalid address. The list of the rejected business objects is then passed to the data cleansing team. With the event activated in a second run, all business objects with invalid addresses are accepted (of course, only when there is no further reason for a rejection due to an error in the data).

7.4 FAQ Migration of Business Partner

7.4.1 How to migrate addresses of a business partner?

Problem

How can addresses of a business be migrated? How can multiple addresses and multiple communication data per address be migrated?

Solution

Addresses of a business partner are migrated using the `BUT020` and `BUT021` structures of the automation data of the `PARTNER` or `PARTNERCHA` migration objects. In the `BUT020` structure, the address itself is transferred and the `BUT021` structure the address usage is transferred (for example, to mark an address as the standard address). The configuration and usage of these structures is explained in the following chapter.

Structure `BUT020`

You can transfer with the `BUT020` structure addresses and address dependant communication data, such as, telephone data. The `BUT020` structure contains a set of control fields, whose names start with `CHIND_`. These fields are the change indicators to pass the information to the application. For example, what exactly the application should do with the data that corresponds to the change indicator.

To create a new address for a business partner the `CHIND_ADDR` field must be populated with an `I` (= Insert) beside the fields that hold the address itself (for example, `CITY1` and `STREET` fields). If you need to migrate telephone data, the respective fields for telephone data have to be populated (for example, `CHIND-TEL` = `'I'` or `TEL_NUMBER`). If more than one telephone number needs to be created for one address, the `BUT020` structure is transferred with all the address data fields set to space, and the fields for telephone data are populated as for the first telephone data (for example, `CHIND-TEL` = `'I'` and `TEL_NUMBER`). One telephone data of all telephone data per address has to be marked as default (`TEL_DEFLT` field). The procedure described for telephone data is also valid for any other communication data that you create, such as, FAX and email (SMTP).

Figure 7-5 shows an example of how to transfer address data for a business partner who has three addresses (number 1 through 3):

- Address 1 - being the standard address - having three different telephone numbers with number 1c being the default number and number 1b being a mobile number
- Address 2 - having two different telephone numbers with number 2a being the default number and also a mobile number
- Address 3 - has no telephone numbers associated

CHIND_ADDR	Address Data	CHIND_TEL	Telephone Data	TEL_DEFLT	TEL_MOBILE
'I'	<address 1>	'I'	<telephone 1>		
		'I'	<telephone 2>		3
		'I'	<telephone 3>	'X'	
'I'	<address 2>	'I'	<telephone 1>	'X'	
			<telephone 2>		3
'I'	<address 3>				

Figure 7-5: Creation of Multiple Addresses of a Business Partner

The `ADEXT_ADDR` (address number in external system) field is a unique identification of an address. You use each value once for all `BUT020` data records of a business partner. Its use

is optional (see below). If business partners with several addresses are transferred, this field can be used to differentiate these addresses. This field becomes mandatory if one of these addresses is either referenced in another business object (for example, in a contract account), the address is subject to change in a later migration step, or is referenced while creating the address (for example, allocating an address usage to an address). The field must be populated with unique information for each business partner.

Structure BUT021

Amongst all addresses transferred in the BUT020 structure, only one can be the standard address. The standard address has to be determined using the BUT021 structure. Otherwise, the first transferred address becomes the standard address by default. Use the ADEXT_ADVW field to identify one of the addresses that the address usage is to be assigned to. There must be matching information transferred in ADEXT_ADDR (address number in external system) field of the BUT020 structure of one of the transferred addresses.

The shown sequence of BUT020 structures in Figure 7-5 can be followed by a BUT021 structure that identifies the standard address. Because the standard address is the first address, the BUT021 structure does not need to be transferred. Assuming that the second transferred address is the standard address, the usage of the BUT021 structure becomes mandatory and you must populate the BUT020-ADEXT_ADDR field.

7.4.2 How to change a migrated address of a business partner

Problem

In some migration scenarios, the addresses of a business partner must be changed. How can this be done with the PARTNERCHA migration object?

Solution

Addresses of a business partner can be changed, deleted, or more addresses can be added to a business partner using the PARTNERCHA migration object. In principal, the rules described are the same except that the CHIND_ADDR change indicator must be populated by either of the values M (= Modify), D (= Delete), or I (= Insert).

7.4.3 How to create SD customers while migrating business partner

Problem

The SD customer (customer in the sales and distribution) can be created while creating a business partner. What configuration is required in the PARTNER migration object?

Solution

Fill the MUSTER_KUN field in the INIT structure with the ID of an already existing SD customer and use this as a template. During the creation of the business partner, copy the SD customer and save the copy with the ID of the newly created business partner.



It is possible to create an SD customer for a business partner, for which no SD customer has been created yet. If you fill the PARTNERCHA migration object the MUSTER_KUN field during the migration, the application checks whether a SD customer for this business partner already exists. If not, the SD customer is created from the given template. If a SD customer already exists, an error message is created and there is no change to the business partner.

7.4.4 How transfer data for only fields which are subject to be changed?

Problem

You are adding a new address for a business partner with the PARTNERCHA migration object. The INIT and BUT020 structures are transferred in the import file. The application raises the error 09 171 *Enter the name of BP*. After transferring additionally the BUT000 structure in the import file and transferring values only for the mandatory fields of this structure, the application accepted the. But it cleared all fields in the business partner, which have not been transferred in the BUT000 structure. Is it correct that it is required to re-transfer all data of a structure, even though only a few fields are to be changed?

Solution

The ISU_M_PARTNER_CREATE_DARK service function module of the PARTNERCHA migration object calls the API (application programming interface) BUS_CONTROL_MAIN_DI to change a business partner. Instead of transferring the same data again or reading the data first from the database, the nodata character can be used to indicate which fields should remain unchanged. The default nodata character is /. The nodata character can be changed in the interface of the ISU_M_PARTNER_CREATE_DARK service function module in the interface definition of the PARTNERCHA migration object. This might be necessary, if a real value is the same as the nodata character.

To simplify the application of the nodata character to structure, use the NO_DATA_INITIALIZE function module in a preprocessing code of the BUT000 structure. This will populate all fields of this structure with the given nodata character (in the example, the nodata character /). This avoids a lot of configuration work in the IS Migration Workbench. Figure 7-6 shows the relevant code snippet.

```
CALL FUNCTION 'NO_DATA_INITIALIZE'
  EXPORTING
    i_structure = '$$$'
    i_nodata    = '/'
  changing
    e_data      = $$$.
```

Figure 7-6: Preprocessing Code to Apply the Nodata Character

7.4.5 Error R1 140: Specify at least one valid business partner role

Problem

The load report cancels with a short dump due to the runtime error MESSAGE_TYPE_X. The Error analysis section details the error R1 140 *Specify at least one valid BP role*.

Solution

The cause of the problem is that no value has been transferred in the INIT-BU_RLTYP field. SAP delivers this field as mandatory but the user has unmarked the *mandatory* indicator (**Req.Fld** field) in the field maintenance screen (for more information, see chapter 7.2.5 *Warning EM 188: Mandatory indicator deleted*).

7.4.6 How to migrate identification numbers of business partners?

Problem

What configuration is required of the PARTNER migration object to create identification numbers for a business partner?

Solution

Identification numbers of a business partner are migrated using the BUT0ID structure of the automation data of the PARTNER or PARTNERCHA migration object. The BUT0ID structure contains the CHIND_ID control field. Use this change indicator to pass the information to the application what the application should do with the identification data corresponds to the change indicator. To create a new identification number for a business partner, populate the CHIND_ID field with an I (= Insert) beside the fields which hold the identification information itself (such as, IDNUMBER).



A common question is, whether the identification numbers of all business partners have to be collected and cleansed before the go-live. If the data cleansing team does not succeed in achieving this goal, the related business partners are migrated without an identification number. Would this be a valid approach? The answer to this question is similar to the question related to a cleansing of address data. We do not recommend migrating wrong identification numbers or creating business partners without required identifications numbers. This is due to the aforementioned impact on business processes while changing business partner data. If the data cleansing team can not achieve the goal cleansing all identification numbers, the project should consider using generic identification numbers and implement a front office process, (for example, the call center agent), to signal that the identification number should be validated with the calling customer.

7.4.7 Migration of direct input structures that are not part of the standard?

Problem

You need to migrate a business partner in the UKM000 role (credit management) with data relevant for this role. You cannot find the related direct input structures for credit management in the automation structure of the PARTNER migration object. Why are these structures missing and how can you add them to the automation data?

Solution

The direct input structures of credit management are part of the FINBASIS (financial basis) package. The automation data structure of the PARTNER migration object belongs to the EEMI (IS Migration Workbench) package, for which the FI-CA (contract accounts receivable and payable) package is the surrounding package. The FINBASIS package and the FI-CA package have no further surrounding package in common. Therefore, the direct input structures of credit management cannot be included in the automation data structure of the migration object in the SAP standard delivery.

It would not be sufficient to add (with a modification) the required direct input structures to the ISUMI_PARTNER_AUTO automation data structure. This is because the ISU_M_PARTNER_CREATE_DARK service function module of the PARTNER migration object calls the API (application programming interface) BUS_CONTROL_MAIN_DI, which provides the T_DATA parameter in its parameter interface to transfer the data of all developed direct input structures. Before calling the API BUS_CONTROL_MAIN_DI function module, the service function module converts the data in the automation data structures to the format of the T_DATA parameter. It does this by calling standard function modules developed for the conversion of the structure, which are part of the automation data structure (for example, the BUP_DI_DATA_BANK function module converts the data of the BUT0BK direct input structure).

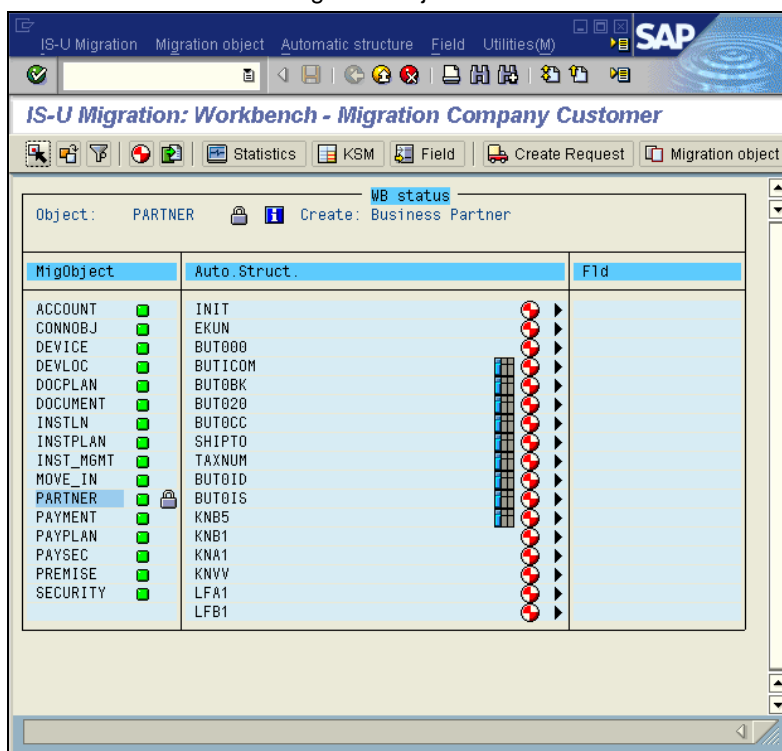
To avoid modifications, the ISUMI_PARTNER_AUTO automation data structure contains a component called T_DATA as internal tables of the same type the API BUS_CONTROL_MAIN_DI request for its T_DATA interface parameter. The service function module adds the information provided in this internal table of the automation data to the list of converted structures (which are part of the standard delivery) before calling the API BUS_CONTROL_MAIN_DI function module. To solve this problem, provide the data for the credit management already fully prepared in the AUTO-T_DATA internal table of the ISUMI_PARTNER_AUTO automation data structure.

Figure 7-7 shows how to add the required additional structures for the *Credit Management* role to the PARTNER migration object. The new structures are as follows:

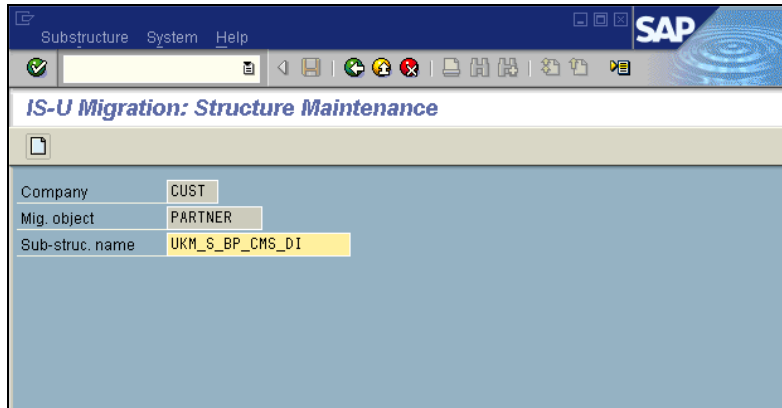
- UKM_S_BP_CMS_DI for score and risk class (structure may occur only once)
- UKM_S_BP_CMS_SGM_DI for credit segment data (structure may occur multiple time)

Configuration steps for UKM_S_BP_CMS_DI structure

1. Double-click the PARTNER migration object to choose it.



2. Choose *Automation Structure* → *Create*
3. Enter the name of the UKM_S_BP_CMS_DI structure and choose *Create*



Substructure System Help

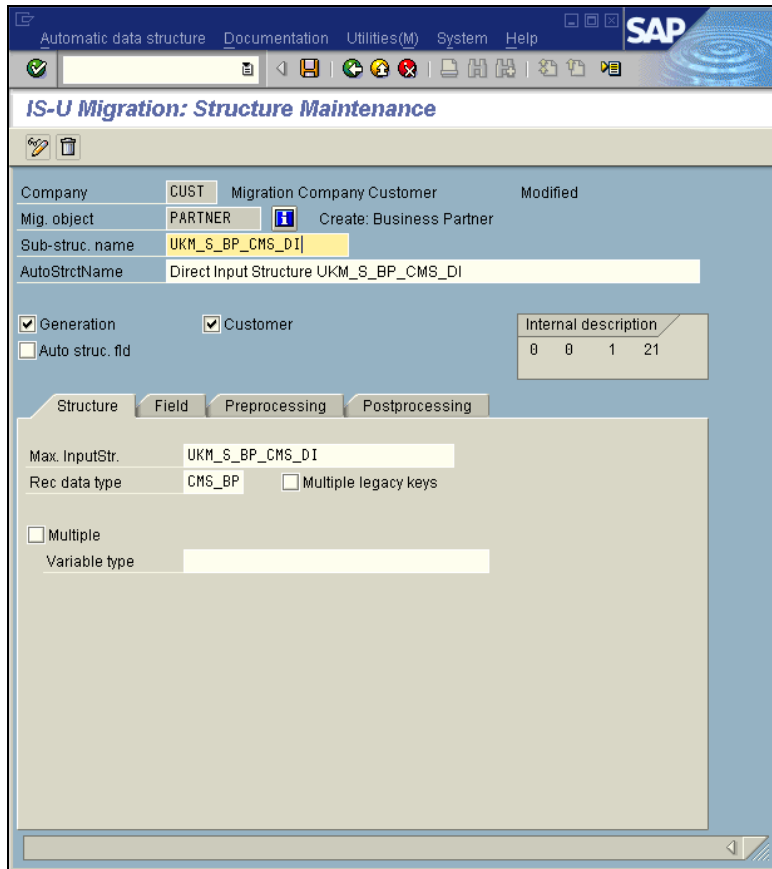
IS-U Migration: Structure Maintenance

Company CUST

Mig. object PARTNER

Sub-struct. name UKM_S_BP_CMS_DI

4. Enter the description of the structure. Enter the name of the UKM_S_BP_CMS_DI structure as **Max. Input Structure**. Mark the **Generation** checkbox. Mark the **Customer** checkbox to indicate that the new structure is not part of the automation structure. Enter a unique **Record Data type** (it must be unique within the PARTNER migration object).



Automatic data structure Documentation Utilities(M) System Help

IS-U Migration: Structure Maintenance

Company CUST Migration Company Customer Modified

Mig. object PARTNER Create: Business Partner

Sub-struct. name UKM_S_BP_CMS_DI

AutoStrctName Direct Input Structure UKM_S_BP_CMS_DI

☒ Generation ☒ Customer

☐ Auto struc. fld

Internal description
0 0 1 21

Structure Field Preprocessing Postprocessing

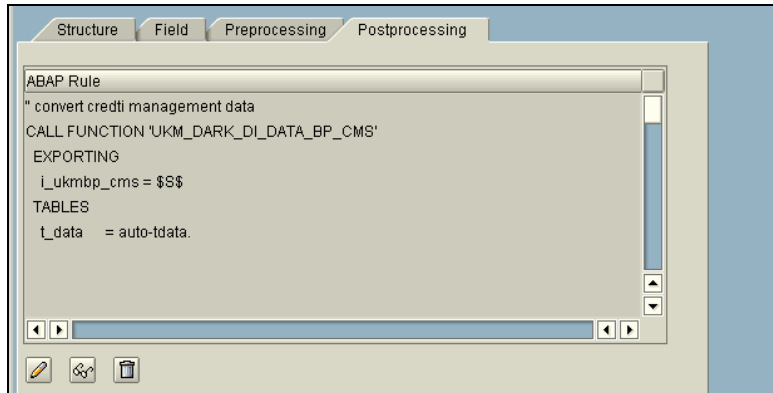
Max. InputStr. UKM_S_BP_CMS_DI

Rec data type CMS_BP ☐ Multiple legacy keys

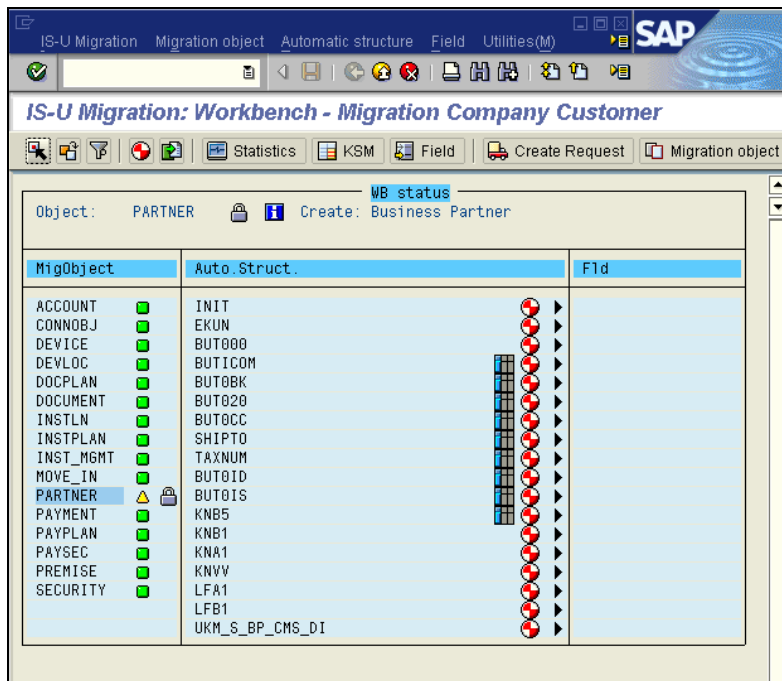
☐ Multiple

Variable type

5. On the **Postprocessing** sub-screen, enter the following ABAP code to convert the data in the structure to the format required by the API BUS_CONTROL_MAIN_DI.



6. Choose **Save**, and return to the main screen to display the added structure.



7. Double-click the **UKM_S_BP_CMS_DI** structure, choose *Automation Structure* → *Compare with DDIC*. After confirming the popup displaying the fields of the **UKM_S_BP_CMS_DI** structure, maintain the field rules of the **UKM_S_BP_CMS_DI** structure.

MigObject	Auto. Struct.	Fld
ACCOUNT	INIT	ALTERNATE_3
CONNOBJ	EKUN	CHECK_RULE_3
DEVICE	BUT000	CREDIT_GRO_3
DEVLOC	BUTICOM	LIMIT_RULE_3
DOCPLAN	BUT0BK	OWN_RATING_3
DOCUMENT	BUT020	OWN_RATING_3
INSTLN	BUT0CC	RATING_CHG_3
INSTPLAN	SHIPTO	RATING_VAL_3
INST_MGMT	TAXNUM	RISK_CLASS_3
MOVE_IN	BUT0ID	RISK_CLASS_3
PARTNER	BUT0IS	RISK_CLASS_3
PAYMENT	KNB5	XCALC_OWN_3
PAYPLAN	KNB1	
PAYSEC	KNA1	
PREMISE	KNVV	
SECURITY	LFA1	
	LFB1	
	UKM_S_BP_CMS_DI	

8. Repeat steps one to four for the UKM_S_BP_CMS_SGM_DI structure. On the structure maintenance screen, mark the **Multiple** field to indicate that this structure is a re-occurring structure for one business partner.

Company: CUST Migration Company Customer Modified

Mig. object: PARTNER Create: Business Partner

Sub-struct. name: UKM_S_BP_CMS_SGM_DI

AutoStrctName: Direct Input Structure UKM_S_BP_CMS_SGM_DI

☒ Generation ☒ Customer

☐ Auto struc. fld

Internal description: 0 0 1 22

Structure Field Preprocessing Postprocessing

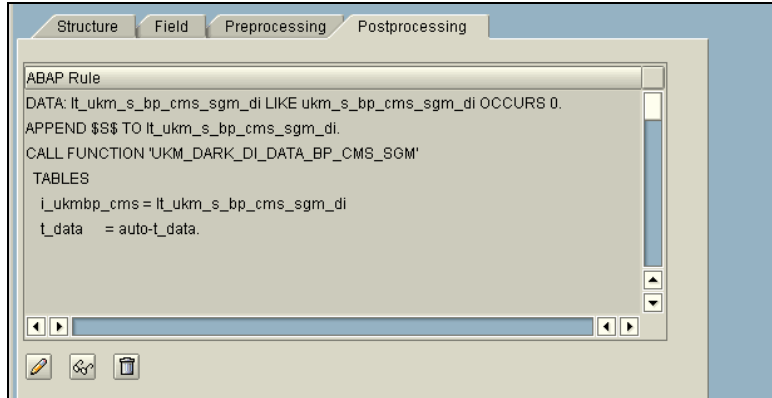
Max. InputStr. UKM_S_BP_CMS_SGM_DI

Rec data type CMS_SG ☐ Multiple legacy keys

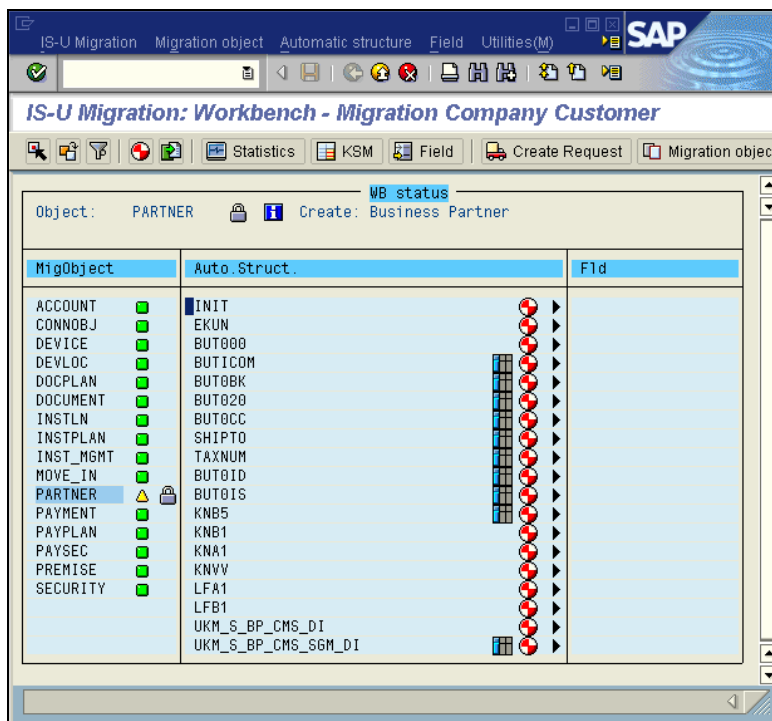
☒ Multiple

Variable type

9. On the **Postprocessing** sub screen, enter the following ABAP code to convert the data in the structure to the format required by the API BUS_CONTROL_MAIN_DI. For the format conversion, the UKM_S_BP_CMS_SGM_DI structure must be passed to the related UKM_DARK_DI_DATA_BP_CMS_SGM function module in the I_UKMBP_CMS internal table. Therefore, before calling the function module, the structure is added to an internal table.



10. Choose Save and return to the main screen to display the added structure.



11. Double-click the UKM_S_BP_CMS_SGM_DI structure, choose *Automation Structure* → *Compare with DDIC*. Confirm the popup displaying the newly added fields the fields of the UKM_S_BP_CMS_SGM_DI structure and maintain the field rules of the UKM_S_BP_CMS_SGM_DI structure.

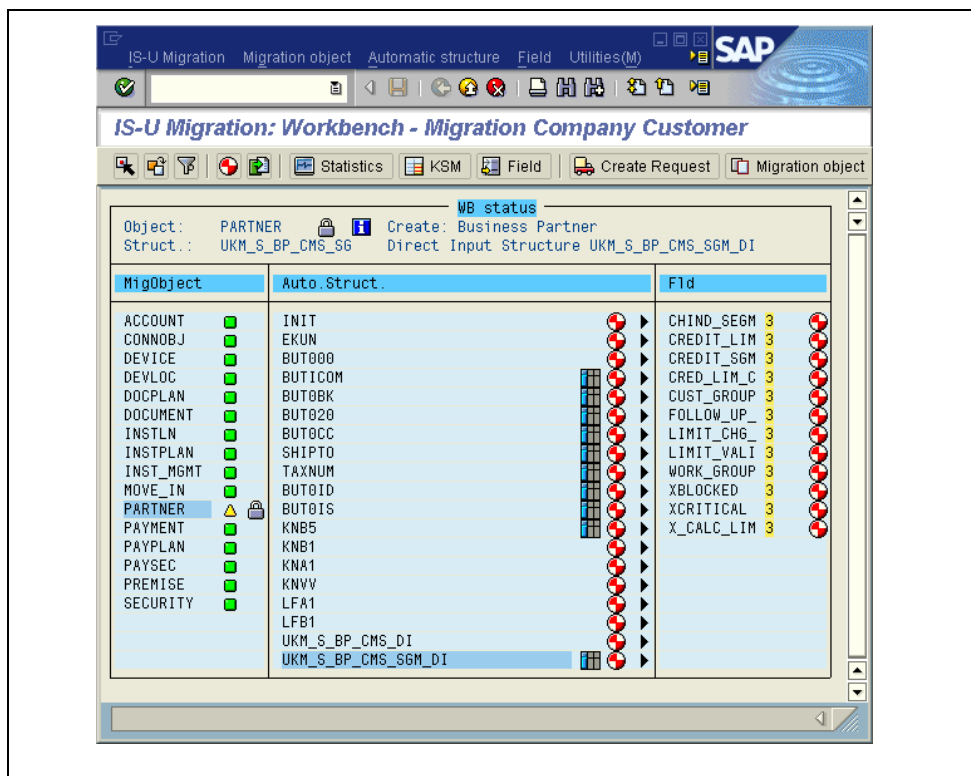


Figure 7-7: Adding a Structure for Credit Management to Migration Object PARTNER

7.4.8 Error AM 890: Internal Error - Value range of LV_DEF_CNR

Problem

The load report for the migration of partner relationships using the PART_REL migration object cancels with a short dump due to the MESSAGE_TYPE_X runtime error. The section *Error analysis* of the short dump details the error with AM 890 *Internal error - value range of LV_DEF_CNR*.

Solution

The BUS052-R_3_USER field is transferred with the value X. The business partner relationship can be migrated without any problems when you are not transferring any value in this field (**Initial Value** field rule). This leaves the field blank. Transfer the value 3 if a mobile number is to be allocated for the relationship.

7.5 FAQ Migration of Contract Account

7.5.1 Error >3 047: Specify bank details for incoming payment methods

Problem

You are creating a second contract account for a business partner who already has a contract account allocated with an incoming payment method specified. The application rejects the new contract account and raises the error message >3 047 *Specify bank details for incoming payment method*. The application uses information (in this case, the bank details of the already existing contract account) and expects the new contract account to specify an incoming payment method.

Solution

A value / must be transferred in the VKP-EBVTY field. The existing bank details are ignored and the contract account is created without an incoming payment method as requested.

7.5.2 Error >3 399: Specify a standard company code of contract accounts

Problem

The application raises the error >3 399 *Enter a value in the field Standard Company Code of Contract Account*. A respective field in the migration Customizing cannot be identified.

Solution

The correct company code must be specified in the VKP-STDBK field. If this field cannot be found a comparison with the ABAP dictionary (DD) must be executed for the VKP structure. Mark the VKP structure by a double-click and choose *Auto structure → Compare with DDIC*. For more information, see 7.2.1 *One or more fields cannot be found in a migration object*.

7.5.3 Error >0 425: Field DFKKLOCKS-LOTYP is not ready for input

Problem

The application raises the error >4 425 *Field DFKKLOCKS-LOTYP is not ready for input -> No update is possible*. This happens only if an incoming payment lock or a dunning lock has been specified and transferred.

Solution

You have specified the lock object category in VKLOCK-LOTYP_KEY field as 04 *Account*. The correct value is 06 *Partner Account Data* because the business locks in the contract account are created for the business partner and not to just for the contract account.

7.5.4 Error >1 004: Administrative data not defined for application T

Problem

The application raises the error >1 004 *Administrative data not defined for application x in company code y*.

Solution

A value has been transferred in the VKP-KZABSVER field (Activate Budget-billing Procedure). You may only use this field in a data migration for the industry solution Utilities and not for the industry solution Telecommunication. Moreover, for all industry solutions, except for Utilities, leave this field blank.

7.5.5 How transfer data for only fields which are subject to be changed?

Problem

You are changing a contract account using the ACCOUNTCHA migration object. The application clears all fields of a structure, even a single field of this structure contains a value to be changed, in the already existing contract account. How can you avoid this?

Solution

The ISU_M_ACCOUNT_CREATE service function module of the ACCOUNTCHA migration object calls the API (application programming interface) BUS_CONTROL_MAIN_DI function module to change a contract account. Instead of transferring the same data again or reading the data first from the database, you can use the nodata character to indicate which fields should remain unchanged. The default nodata character is /. The nodata character can be changed in the interface of the ISU_M_ACCOUNT_CREATE service function module in the interface definition of the ACCOUNTCHA migration object. This may be necessary, if a real value is the same as the nodata character.

In order to simplify the application of the nodata character to structure, the NO_DATA_INITIALIZE function module can be used in a preprocessing code (for example, in the VKP structure). This will populate all fields of this structure with the given nodata character (in the example the nodata character /). This avoids a lot of configuration work in the IS Migration Workbench. Figure 7-8 shows the relevant code snippet.

```
CALL FUNCTION 'NO_DATA_INITIALIZE'  
  EXPORTING  
    i_structure = '$$$'  
    i_nodata    = '/'  
  changing  
    e_data      = $$$.
```

Figure 7-8: Preprocessing Code to Assign the Nodata Character

7.6 FAQ Migration of Financial Documents

7.6.1 Error >0 005: No free number range found for document type

Problem

The application raises the error >0 005 *No free number range found for document type x*. This error usually occurs during the migration of open items (DOCUMENT migration object) or payments (PAYMENT migration object).

Solution

You have scheduled more jobs for the migration of financial documents (for example, open items or payments) than number of range intervals for mass processing are available. Figure 7-9 shows the IMG configuration path to define and maintain number range intervals for financial documents and to assign them to document types.

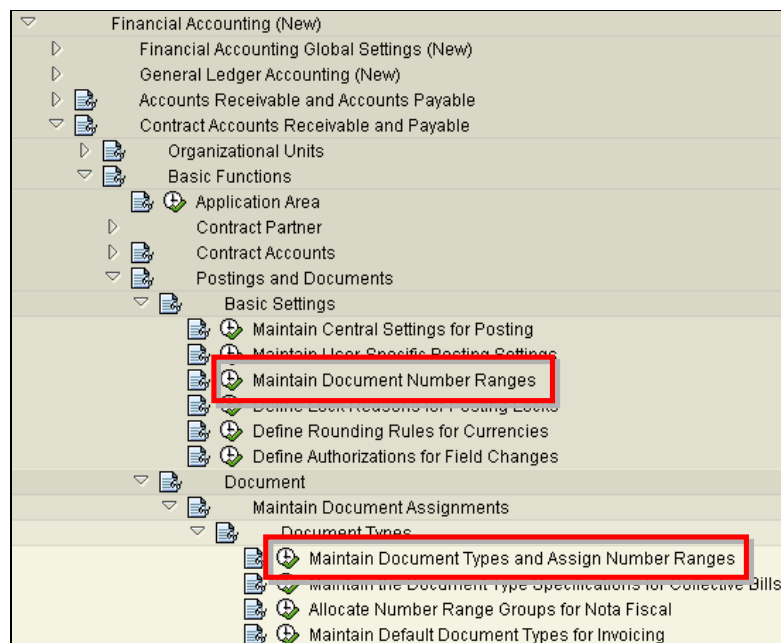


Figure 7-9: IMG Configuration Path

7.6.2 Error >0 200: Posting period for company code already closed

Problem

The application raises the error >0 200 *Posting period x for company code y already closed*.

Solution

Part of the data migration strategy for financial documents can be to migrate financial documents with the same posting date as the documents that have been posted in the legacy system. This means that the documents are to be posted with a posting date of the past years. To allow a posting date, the respective posting periods have to be (re-)opened (transaction OB52).

7.6.3 Is it possible to create installment plans with external numbering?

Problem

You need to migrate installment plans with same number as the legacy system number of the active installment plans. When we transfer the legacy system number in the INTERFACE-OPBEL field the application raises the error >0 036 *Document type x only allows*

document numbers from y to z. How can we migrate installment plans with an external number?

Solution

You cannot migrate an installment plan with an external number because the application does not support this process. This is also true for the `FPR1` dialog transaction. The usage of the `INTERFACE_OPBEL` field is not permitted hence this field is marked as a non-migration field.

7.6.4 How to migrate rounding balances?

Problem

You need to migrate rounding balances that have been calculated in the legacy system. How can rounding balances be migrated?

Solution

There is neither a standard migration object nor a BAPI function module available to migrate rounding balances. There exists the `FKK_ROUND_CREATE` function module to save a rounding balance. However, you cannot use this function module as a service function module of a BAPI migration object because the definition of its parameter interface does not comply to the design rules of a BAPI (for more information about the design rules, see chapter 3.4.2 *Generation of a BAPI Migration Object*). Therefore, you need to develop an own service function module that calls the `FKK_ROUND_CREATE` function module and manage the error handling. Figure 7-19 shows an ABAP code that can be used as a basis for an own development.

```

FUNCTION zzmig_fkk_round_create.
  * " -----
  * " Interface local:
  * " IMPORTING
  * "   REFERENCE(X_FKKRD) TYPE DFKKRD
  * "   REFERENCE(X_UPDATE_TASK) TYPE BOOLE_D DEFAULT SPACE
  * " EXPORTING
  * "   REFERENCE(RETURN) LIKE BAPI_RET2 STRUCTURE BAPI_RET2
  * " -----

  * Local data definition
  DATA: wa_fkkrd TYPE dfkkrd.

  * check for existing rounding balance
  SELECT SINGLE * FROM dfkkrd INTO wa_fkkrd
    WHERE gpart = x_fkkrd-gpart
    AND   vkont = x_fkkrd-vkont
    AND   bukrs = x_fkkrd-bukrs
    AND   waers = x_fkkrd-waers.

  IF sy-subrc = 0.
    CALL FUNCTION 'BALW_BAPI_RETURN_GET2'
      EXPORTING
        type      = 'E'
        cl        = 'EM'
        number    = '000'
        par1      = 'Error updating DFKKRD Entry for BP'
        par2      = x_fkkrd-gpart
        par3      = 'already exists.'
      IMPORTING
        return    = return.
    RETURN.
  ENDIF.

  * create rounding balance (no further error handling required)
  CALL FUNCTION 'FKK_ROUND_CREATE'
    EXPORTING
      i_fkkrd      = x_fkkrd
      i_update_task = x_update_task.

ENDFUNCTION.

```

Figure 7-10: Service Function Module to migrate Rounding Balances

After the generation of the BAPI migration object, change the **Return Field** on the migration object maintenance screen to `AUTO-X_HEAD-TDNAME`. This saves the number of the function location in the KSM.



Change the field rule of the `X_FKKRD-GPART` and `X_FKKRD-VKONT` fields to the field rule 'Via KSM' to ensure that the rounding amount is related to a migrated business partner and contract account.

7.7 FAQ Migration of Open Item

7.7.1 Error >0 009: Reconciliation key already in use

Problem

The application raises the error >0 009 *Reconciliation key x already in use: Choose another key*.

Solution

This error usually occurs, if one of the following conditions is met:

- The reconciliation key is transferred in the import file for the `FKKKO-FIKEY` field
- The same name of the reconciliation key is generated more than once

The background of the problem is that each job migrating open items requires an own unique reconciliation key. In addition, the load programs close the reconciliation key after its use when finishing the import file. It is not a viable option transferring the reconciliation key in the import file. You would have to change the reconciliation key in the import file before reprocessing the original import file or processing the error file. This is not feasible. We recommend generating a new name of a reconciliation key during runtime of a job. For that, the `FIKEY` field is pre-configured with the *Fixed value* field rule calling the `ISU_M_FIKEY_DETERMINE` function module. This function module generates a name for a new reconciliation key based on the information in the `TEMFIKEY` database table. The field rule for this field and the implementation of the `ISU_M_FIKEY_DETERMINE` function module is just a proposal. You can use an own function module with a changed ABAP code to determine unique reconciliation keys per job.

7.7.2 Is it possible to reuse reconciliation keys?

Problem

Is it possible to reuse reconciliation keys when you are migrating documents with the `DOCUMENT` migration object?

Solution

It is possible to reuse reconciliation keys even though this is not recommended. The disadvantage of reusing reconciliation key is that it is more difficult reconcile migrated financial documents between the legacy system and the SAP system. The following changes have to be implemented to use the same reconciliation key more than once:

- In the `DOCUMENT` migration object, the `X_REUSE_FIKEY` parameter of the `ISU_M_DOCUMENT_CREATE` service function module has to be set to X.
- The reconciliation key has to be determined individually. For that, the `ISU_M_FIKEY_DETERMINE` function module in the field rule for `FKKKO-FIKEY` field has to be replaced by an own project specific function module.

For more information about the development of an own algorithm, see chapter 9.4 *Function Module to Determine Reconciliation Keys for a Reuse*.

7.7.3 Migrating open items with division but without contract information?

Problem

The DOCUMENT migration object contains the FKKOP-SPART (division) and FKKOP-VTREF (contract reference) fields. Is it permitted to migrate open items into an SAP system transferring only the division but without a reference to a contract? If the answer were yes, it would not be required to migrate all the technical objects and the contract for customers with only inactive contracts but with the debit on the contract account due to an unpaid bill.

Solution

Technically, it is possible to create documents with a division but without a reference to a contract. The application uses only the division information for the account determination in the FI general ledger. On the other hand, the created data is not fully consistent from a functional point of view. However, there are no negative impacts known related to this approach.

7.7.4 Migrating open items with VAT tax amounts

Problem

Do we need to transfer tax items with value added tax (VAT) amounts when creating invoices (open items) with the DOCUMENT migration object? A migration of invoices with VAT tax amounts may create problems with the tax reporting because the same invoice is posted a second time and leads, for example, to a second declaration of taxes.

Solution

SAP recommends to implement one of the follow solutions:

Method 1 (posting of open items and payables to migration accounts without VAT):

You transfer invoices with the correct VAT tax indicator (FKKOP-MWSKZ field) but without tax general ledger line items. The amount of the revenues in the general ledger line items is the gross amount and corresponds to the amount of the corresponding open item amount. In addition, you must implement the FI-CA event 0070 to prevent a reversal of such documents due to the missing tax amounts. Using the document origin key (FKKKO-HERKF field) in the document header or additional customer-specific fields, your check function module can identify migrated documents and determine whether only a special reversal transaction is permitted for the document in question (use the FQEVENTS transaction to display a detailed documentation of this event). Instead of reversing an open item you must offset the open item with transaction FPE1 using the related VAT tax indicator and suitable transactions. After the offsetting posting, you must carry out the necessary account maintenance to clear the migrated open item by the offsetting document.

Method 2 (posting of open items to migration accounts with VAT):

You transfer invoices with VAT tax indicator (FKKOP-MWSKZ field) and with tax general ledger line items exactly as the open item was posted in the legacy system. To prevent a second declaration of the same tax amounts, you have to make an adjustment posting to an account of a dummy business partner (migration business partner) to correct the total amount of taxes to zero. The total amount is the amount of taxes you have posted during the data migration of the related open items. The adjustment posting requires an additional receivable (reconciliation) and a revenue account in the general ledger. The VAT tax settlement account remains the same to clear the VAT tax amounts that have been posted during data migration.



There are some countries, for example South Africa, where the VAT tax must be declared only after the payment of the invoice. In such situations, you must migrate invoices with the correct tax amounts.



In case of down payments, you must supply the FKKOP-SBETH and FKKOP-SBETW fields. Down payments should always be posted with VAT and further activities are required as described in method 2.



You must plan and execute a comprehensive reconciliation of the financial data on the legacy system before you migrate financial documents into the SAP system. In addition, you must create migration specific VAT settlement account, receivable and revenue accounts in dependency of the chosen method.

7.7.5 Can migrated financial documents be reversed?

Problem

Is it possible to reverse migrated financial documents, such as open items?

Solution

In most cases the answer is that it is not possible to reverse migrated financial documents. This is because they have been migrated without tax amounts. Therefore, a reversal of the tax amount would not be possible.

There are some countries (for example, South Africa) where the VAT is not due during invoicing but during the payment of the invoice. In this case, a reversal of an open item is possible.

7.7.6 Are there any migration specific FI-CA events?

Problem

Are there migration specific FI-CA events available that are executed during data migration?

Solution

The FI-CA events 3910 and 3920 are migration specific and are defined in the transaction `FQEVENTS`. You can use the `FQEVENTS` transaction to display a detailed documentation of these events.

7.8 FAQ Migration of Payment

7.8.1 Why do migrated payments not clear open items?

Problem

With the PAYMENT migration object, it is possible to migrate payments to clear predetermined open items. This allows the migration of a payment history of a contract account as it is in the legacy system. Why do migrated payments not clear any open item?

Solution

There are various reasons why a clearing process does not result in cleared open items. The main causes are as follows:

- Wrong or incomplete selection of open items to be cleared
- The selected open items are already cleared
- Wrong or incomplete setup of the clearing control and clearing rules

The first two causes are related to problems during the extraction or transformation process and need to be fixed by the data migration team. The responsibility to solve problems in the clearing control is with the functional FI-CA team. Define the clearing control for migrated payments to improve the performance of the data migration of payments. The simplest way to determine whether a payment is a migrated payment is to analyze the information in the `HERKF` field (document origin key). Ideally, a migration specific origin key is used to allow the use of a migration specific clearing variant. This migration specific clearing variant can be kept very simple and allows the clearing of any selected open item of the same contract account.

7.8.2 Error due to competing processes

Problem

During the migration of payments the application raises either the error message `>0 103 Complete selection not possible due to competing processing` or `>0 105 Complete selection not possible due to block by user x`.

Solution

Both error messages are pointing to the same problem and the long texts of the error messages are explaining the root of the problem. The clearing process for one contract account will not be completed if you initiate a second clearing process for the same contract account. This is to prevent a double clearing of the same open item.

It is not an option to repeat the migration of the same import file with migrations until all payments have been migrated. The reduction of the commit buffering value to 1 for the migration payment is not a solution. Even if all business locks on the related business partner and contract account are released with a `COMMIT WORK`, there is still the risk of accessing the same contract account for a clearing process in parallel processes. We do not recommend the release of the business lock in an event right after calling the service function module for the PAYMENT migration object because the very critical clearing process might be compromised.

You can solve the problem in the following ways:

- Execute data migration with a single process per import file with commit buffering 1
- Execute data migration with the *Distributed Import* (data import of the same import file with parallel jobs) but with import files which contain only one payment for the same contract account.

We do not recommend option one due to performance limitations. The second approach allows the use of commit buffering to improve the performance of the data migration of

payments. The downside of the second approach is, that more import files have to be created and migrated. This makes the data migration process more complex. The number of import files should be kept to a minimum. Create a final import file that collects all remaining payments, even for the same contract account. This import file must then be migrated with a single job (with commit buffering set to 1) to avoid the described blocking situations. The data migration team should use the *Migration Lock Mechanism* function to simplify the migration process (for more information, see chapter 2.8).

The described scenario may occur during both the clearing of single opens items and the migration of payments of an installment plan or budget-billing plan.

7.8.3 Error >0 009: Reconciliation key already in use

Problem

The application raises the error >0 009 *Reconciliation key x already in use: Choose another key.*

Solution

This error usually occurs if one of the following conditions is met:

- The reconciliation key is transferred in the import file for the EMIG_PAY_FKKKO-FIKEY field
- The same name of the reconciliation key is generated more than once

For more information about this problem and its solution, see chapter 7.7.1 *Error >0 009: Reconciliation key already in use.*

7.8.4 Is it possible reusing reconciliation keys

Problem

Is it possible to reuse reconciliation keys when you are migrating payments with the PAYMENT migration object?

Solution

It is possible to reuse reconciliation keys but we do not recommend it. The disadvantage of reusing reconciliation keys is that it is more difficult to reconcile migrated financial documents between the legacy system and the SAP system. Implement the following changes to use the same reconciliation key more than once:

- In the PAYMENT migration object the X_REUSE_FIKEY parameter of the ISU_M_PYAMENT_CREATE service function module has to be set to X.
- The reconciliation key is determined individually. For that, the ISU_M_FIKEY_DETERMINE function module in the field rule for the EMIG_PAY_FKKKO-FIKEY field has to be replaced by an own project specific function module.

For more information about the development of an own algorithm, see chapter 9.4 *Function Module to Determine Reconciliation Keys for a Reuse.*

7.8.5 Is it possible to use the DOCUMENT migration object for payments?

Problem

Is it possible to migrate payments as credits with the DOCUMENT migration object?

Solution

It is possible to migrate payments as credits. However, a subsequent clearing process has to be implemented as part of the data migration. The clearing is done by executing a mass activity (transaction *FPMA Automatic Clearing*) as an additional step. The disadvantage is that the clearing variants have to be defined to deal with migrated historical payments, which leads to more complex clearing variants.

7.8.6 Error >4 036: “Document does not exist” when paying security deposits

Problem

The PAYMENT migration object is used to pay the open requests of migrated security deposits. The error message >4 036 *Document x does not exist* is raised for all payments so none of the payments can be migrated. What has to be changed to pay the request document of a cash security deposit?

Solution

When you migrate a cash security deposit the *security object* business object and an FI-CA request document are created. Pay the request document with the PAYMENT migration object. The PAYMENT migration object is configured to transfer the oldkey of the security object in the EMIG_PAY_FKKKO-OIBEL field; and the EMIG_PAY_FKKKO-OIBEL field has been configured with the *via KSM* field rule and the SECURITY top object. This configuration does not work as the SECURITY migration object is configured by default to save the key of the created security object in the KSM. The error > 036 *Document x does not exist* is raised because the application searches the request document with the key of the security object.

The key of the security object is saved in the KSM according to the configuration in the **Return** field (FKKSE_SEC-G-SECURITY) of the SECURITY migration object on the migration object maintenance screen. The problem is that a request document number is required for the payment and not the number of the cash security deposit.

You can solve the problem in the following ways:

- Add the code of Figure 7-11 to the postprocessing code of the EMIG_PAY_FKKKO structure (assuming that you use the EMIG_PAY_FKKKO-OIBEL field to transfer the oldkey of the cash security deposit). The ABAP code replaces the newkey of the security deposit by the request document number that it can be found in the OPBEL field of the FKK_SEC_REQ database table.

```


if not $$-oibel is initial.
  select single opbel from fkk_sec_req
                        into $$-oibel
                        where security = $$-oibel .

  if sy-subrc ne 0.
    clear $$-oibel .
  endif.
endif.

```

Figure 7-11 Postprocessing Code for a Payment for a Request Document

- Replace FKKSE_SEC-G-SECURITY with FKKSE_SEC-G-OPBEL in the field **Return Field** on the **Migration Object Maintenance screen** for the SECURITY migration object. The request document number of the security deposit, instead the number of the security deposit itself, is now saved in the KSM when migrating security deposits. The *via KSM* field rule in the PAYMENT migration object will now return the required request document number of the security deposit when transferring the oldkey of the security deposit in the import file. Additionally, change the migration class of the SECURITY migration object from SECURITY to DOCUMENT. Figure 7-12 shows the changed Customizing for the SECURITY migration object.

Company	CUST	Migration Company Customer
Mig. object	SECURITY	
Mig. obj. text	Create: Security Deposit	
Mig. obj. abbr.	SEC	3 Create
Blocking Status	000	Data import blocked

Object Data	Relationship data	Parameters
-------------	-------------------	------------


Service module	FKK_S_SECURITY_CREATE	 Interface
Auto struc.type	FKKSE_SECURITY_AUTO	<input type="checkbox"/> Mult. leg. keys
Return structure	FKKSE_SEC	<input checked="" type="checkbox"/> Structure type
<input type="checkbox"/> Internal Table		FKKSE_SEC
		<input type="checkbox"/> Structure type
Return field	FKKSE_SEC-G-OBPEL	

Figure 7-12 Maintenance Screen of the SECURITY migration Object

7.9 FAQ Meter Reading

7.9.1 Creating move-in meter reading during device installation

Problem

During the migration of a billing related installation of a device with the INST_MGMT migration object, a meter reading is created with meter reading reason 21 (billing related installation) and with reason 06 (move-in). The reason 06 is created even no contract exists and will not exist at the device installation date. This creates problems during billing. How can the creation of the meter reading reason 06 be suppressed?

Solution

In the standard migration scenario, the device installation date and the move-in date is the same. For more information about the standard migration scenario, see chapter 6.2.1 *Standard Scenario (No Installation Structure History)*. During a billing related installation of a device a meter reading reason 06 (move-in) is created automatically. This is because it is not possible to migrate second a meter reading with meter reading reason 06 (move-in) for a date for which a meter reading already exists. The automatic creation serves both functional and performance needs. The automatic creation of the meter reading reason 06 (move-in) can be suppressed. For this, transfer the CONTAINER structure in the import file with the NO_AUTOMOVEIN field set to X.

7.9.2 Creating different meter reading types during device installation

Problem

Is it possible to use during a device installation with the INST_MGMT migration object a different meter reading type than the default meter reading type *Meter Reading by Utility*?

Solution

It is not mandatory to transfer a meter reading when migrating a device installation (for further information, see chapter 7.9.5 *Migration of meter reading required during device installation/removal?*). When using the METERREAD migration object, the meter reading type can be set explicitly.

7.9.3 Billing Error “Move-in meter reading missing or incorrect”

Problem

Billing terminates with message *Move-in meter read. result missing or incorrect*. The application does not recognize that the move-in reading has already been updated to the move-in document, which then causes a billing error.

Solution

During the migration of move-in (MOVE_IN migration object), no meter readings are created for performance reasons. Create the required meter reading with the meter reading reason 06 either during device or by using the METERREAD migration object. If the device is installed with a date after the move-in date, the creation of the meter reading reason has to be suppressed. However, you can create a meter reading order during the migration of a move-in in using the MOI_MR_ORDER_OFF migration control parameter (for more information, see chapter 3.6.2 *Control Parameters Specific to SAP for Utilities*)

7.9.4 Creating meter reading order?

Problem

How can we migrate meter reading orders?

Solution

You need supply the following fields of the METERERAD migration object:

Field Name	Description
IEABLU-EQUNR	Equipment number
IEABLU-ADAT	Meter reading date relevant to billing
IEABLU-ABLESEGR	Meter reading reason

You must not supply the following fields of the METERERAD migration object (clear the field **Required Field** for these fields):

Field Name	Description
IEABLU-ZWSTAND	Meter reading
IEABLU-ISTABLART	Meter reading type
IEABLU-ABLESER	Meter reader

In addition, change the `X_NO_READ_ORDER` parameter in the interface of the `ISU_M_METERREAD` service function module of the METERREAD migration object from 'X' to `SPACE`. For further information, see SAP notes 1499017 and 1500736.

7.9.5 Migration of meter reading required during device installation/removal?

Problem

A meter reading is required when migrating a device installation, removal and replacement and must be supplied in the `AUTO_ZW-ZWSTANDCE` and `AUTO_ZW-ZWSTANDCA` fields of the automation data of the `INST_MGMT` migration object. Is it possible to migrate the required meter readings afterwards with the METERREAD migration object?

Solution

It is not mandatory to supply a meter reading when migrating a device installation, removal and replacement. A meter reading is only required if `SPACE` is passed (standard) in the `X_NO_MR_REQUIRED` parameter of the parameter interface of the `ISU_S_WORKLIST_INSTALL` service function module of the `INST_MGMT` migration object. If an 'X' is passed, no meter reading is required and can be migrated afterwards in using the METERREAD migration object.

7.9.6 Deferred migration of meter reading for a register relationship?

Problem

A meter reading is required when creating a register relationship using the `REGRELSHIP` migration object. Is it possible to migrate the required meter readings not before but afterwards with the METERREAD migration object?

Solution

It is not mandatory to supply a meter reading first when migrating a register relationship. A meter reading is only required if the `EG75_NOMR` control parameter is not used. Figure 7-13 shows the IMG configuration path.

7.9.7 Deferred migration of meter reading for rate data changes of devices?

Problem

A meter reading is required when creating and changing the rate data of devices using the DEVICERATE migration object. Is it possible to migrate the required meter readings not before but afterwards with the METERREAD migration object?

Solution

It is not mandatory to supply a meter reading first when create and change of rate data of a device. A meter reading is only required if the `EG70_NOMR` control parameter is not used. Figure 7-13 shows the IMG configuration path.

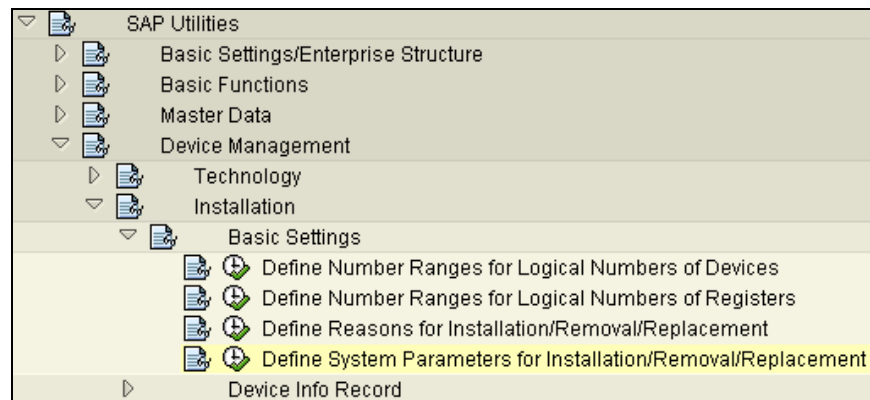


Figure 7-13: IMG Configuration Path

7.10 FAQ Creation of Inactive Contracts

7.10.1 How suppress billing triggers when migrating inactive contracts

Problem

During the migration of inactive contracts with the MOVE_IN_H migration object, the application creates a billing trigger for each contract, even though the value X is transferred for the FAKTURIERT field. How can the creation of the billing trigger be suppressed?

Solution

A move-out always creates a billing trigger because the migrated contract could be a move-out without a final billing in the legacy system. Therefore, the application creates a billing trigger, as well as a meter reading order for the installation. The standard scenario foresees a subsequent migration of a billing document (for example, BILLDOCEBF migration object). During migration of a billing document, the application compares the end-of-billing-periods of the billing documents with the move-out date of the related contract. If these dates match, the application deletes the created trigger and updates the FAKTURIERT and BILLFINIT fields. The SAP system delivers the FAKTURIERT field as a non-migration field, and so, it cannot be used during data migration. Figure 7-14 shows an ABAP code snippet that deletes the created billing trigger directly in the database if a billing document is not to be migrated with a matching end-of-billing-period date for the inactive contract. The code snippet can be added as an event at the CALL02 generation point.

```
IF sy-subrc = 0 AND db_update = 'X'.
  DELETE FROM ETRG WHERE anlage = everd-anlage
                        AND abrdats = everd-ausdat
                        AND abrvorg = '03'.
ENDIF.
```

Figure 7-14: Event to Delete Billing Trigger for Move-Out

7.10.2 Error AH385: Contract must be billed first?

Problem

An installation cannot be billed due to the error message AH385 *Installation x: Contract y must be billed first*. The error long text says that it is not possible to bill the contract because the contract y was already allocated to the installation and final billing has not been executed for the contract y. The contract y has been migrated with the MOVE_IN_H migration object.

Solution

The MOVE_IN_H migration object does not mark the contract as billed in updating the FAKTURIERT and BILLFINIT fields. Either an EBF billing document has to be migrated with an end-of-billing-period that matched the move-out date (recommended), or the contract has to be updated during data migration. Figure 7-15 shows an ABAP code snippet that updates the contract directly in the database. The code snippet can be added as an event at the CALL02 generation point.

```
IF sy-subrc = 0 AND db_update = 'X'.
  LOOP AT iever INTO j_ever.
    UPDATE EVER WHERE vertrag = j_ever-vertrag
      SET FAKTURIERT = 'X'
          BILLFINIT = 'X'.
  ENDLOOP.
ENDIF.
```

Figure 7-15: Event to Mark a Contract as Billed

7.10.3 Creating meter reading orders with reason move-out?

Problem

During the migration of inactive contracts with the MOVE_IN_H migration object, the application creates a meter reading order for each contract. How can the creation of the meter reading order be suppressed?

Solution

A move-out always creates a meter reading order if a device is installed because the migrated contract could be a move-out without a final billing in the legacy system. Therefore, the application creates a meter reading order for the device. A subsequent migration of meter readings (METERREAD migration object) will pickup the order and updates it with the transferred meter reading. Figure 7-16 shows an ABAP code snippet that deletes the created meter reading order directly in the database if a meter reading is not to be migrated for the move-out date. The code snippet can be added as an event at the CALL02 generation point.

```
IF sy-subrc = 0 AND db_update = 'X'.
  DATA: lv_abl bel nr type eabl g-abl bel nr.
  SELECT abl bel nr FROM eabl g INTO lv_abl bel nr
        WHERE anlage = everd-anlage
              AND adatsol1 = everd-ausdat.

  IF sy-subrc = 0.
    DELETE FROM eabl g WHERE abl bel nr = lv_abl bel nr.
    DELETE FROM eabl WHERE abl bel nr = lv_abl bel nr.
  ENDIF.
ENDIF.
```

Figure 7-16: Event to Delete Meter Reading Order for Move-Out

7.10.4 Creating active contracts with the MOVE_IN_H migration object?

Problem

Is it possible to simplify the migration of contracts by using only the MOVE_IN_H migration object to migrate inactive as well as active contracts?

Solution

The MOVE_IN_H migration object can create inactive as well as active contracts. The disadvantage of using this migration object for the migration of active contracts is that the MOVE_IN_H migration object is not released for commit buffering, and so, performs worse in comparison to the migration of active contracts with the MOVE_IN migration object (designed to migrate only move-in). The MOVE_IN migration object is enabled for commit buffering.

7.11 FAQ SAP for Utilities

7.11.1 No update of business partner addresses during move-in

Problem

During data migration move-in with the MOVE_IN migration object, the address list of the business partner is not completed by the address of the related premise (connection object), even if an update is configured in the IMG activity *Define move-in control parameters*. In addition, why is no move-in letter created even it is configured in the same IMG activity? A similar question is, why the search index for the business partner service address (EBP_ADDR_SERVICE table) is not updated even it is configured in the IMG activity *Define System Parameters for IS-U*.

Solution

During the migration of a contract with the MOVE_IN migration object, only the application creates only the contract and the move-in documents. No further actions take place. This is due to performance reasons but also because it is not a required function during a data migration. However, you can add the address of the connection object to the address list of a business partner already during the migration of the business partner by transferring the SHIPTO structure of the PARTNER migration object to link the address of the connection object to the business partner.

If you want to use the search index, then you must create the index using the REU_SEARCH_IND_UPDATE report (update search index for business partner by service address) upon completion of the migration of the contracts (MOVE_IN and MOVE_IN_H migration objects). After go-live, the system keeps the index up-to-date.

7.11.2 Migration of multiple time slices for an installation

Problem

You need to migrate multiple time slices for an installation to reflect certain information (for example, rate changes on installation level). How can these time slices be migrated?

Solution

It is not possible to migrate multiple time slices for the same installation in one single data object. This is due to the *BIS (english for TO)* field that occurs only once in the *KEY* structure INSTLNCHA migration object. With this, the changes for one installation have to be migrated chronologically. The from-date of the time slice should be part of the oldkey to avoid duplicate oldkeys being used.

Special attention has to be paid to the error handling when migrating multiple time slices. If the second time slice fails to migrate and the third one can be migrated without a problem, it will not be possible anymore to migrate the second time slice. This is due to the now existing third time slice. The following options are available depending on the number of installation changes to be migrated:

- Migration of the installation changes sequentially and using the lock mechanism of the IS Migration Workbench (for more information, see chapter 2.8 *Migration Lock Mechanism*). It is not necessary to have all installation changes in one single import file. You can create multiple files to be imported in parallel, but one installation should not be changed in parallel files.
- Create as many import files as changes for each installation to be migrated. The first import file holds the first changes for all installations; the second import file holds the second changes and so on. With this, the import file can be imported using the distributed import as no competing changes to an installation can occur with this approach.

7.11.3 The start date of the device group is set to the system date?

Problem

When migrating device groups with the DEVGRP migration object, the start date of the device groups are set to the current date even though the correct date has been transferred in the EMG_EDEVGR- KEYDATE field. How can the correct dates be set for a device group?

Solution

The field Customizing of the EMG_EDEVGR- KEYDATE field has been changed from *ABAP rule* field rule to *Transfer* field rule. The delivered field rules populate the AUTO-KEYDATE field in the automation data and are mandatory.

7.11.4 PRUEFKLE field remains empty after device installation**Problem**

When installing a device with the INST_MGMT migration object (device installation) the validation class (PRUEFKLE field in the AUTO_ZW structure) is not determined from the settings of the related register group.

Solution

For more information about the solution for this problem, see SAP note 168957 *Fields not set at register level*. The problem is that the data migration uses direct input where spaces in the automation data are recognized as spaces by the application. One solution is to transfer the correct value in the automation data. The alternative is to use the customized default value, which is determined automatically by the application during the respective dialog transaction. In the latter case, these fields have to be included in the CI_REG30_ZW_I_SUPPRESS_MIG include structure (defined in REG30_ZW_C standard structure). The application will use the customized value instead of using space as the correct value.

7.11.5 No migration of custom fields for device locations**Problem**

During data migration of device locations with the DEVLOC migration object, the custom fields are not transferred to the IFLOT database table. Why not?

Solution

It is required to implement the enhancement ES650001 (component EXIT_SAPLES65_008) for a transfer of custom fields for a device location. For more information, see chapter 4.4.3 Enhancement Fields for Technical Master Data (SAP for Utilities).

7.11.6 Error AH 348: The facts are not consistent**Problem**

During data migration of installations with the INSTLN migration object the application raises the error AH 348 *The facts are not consistent*. This is because the creation of installations requires the creation of facts marked as required. How is it possible in such a situation to migrate installation facts using the FACTS migration objects?

Solution

The INST_FACTS_REQ_OFF control parameter controls the handling of the required operand indicator during the migration of installations (INSTLN migration object) or changes to installations (INSTLNCHA migration object). Setting this parameter to X has the effect that the application ignores missing required values for operands when creating or changing installations. Installation facts have to be migrated in a subsequent step using the FACTS migration object.

7.11.7 Error M3 897: Plant data of material is locked

Problem

During data migration of goods movement with the GOODSMVT migration object (or a generated migration object using the `BAPI_GOODSMVT_CREATE` function module the application raises the error M3 897 *The plant data of the material x is locked by the user y*. How can this situation be avoided?

Solution

There are only few ways to avoid this error message. For more information, see SAP notes 322989 *Late block: Number of blocking attempts* and 70865 *Material block during goods movements*. You have the following options:

- Reduce the number of parallel import jobs
- Increase temporarily the time-out period
- Run data imports with a single job per material (for more information, see chapter 7.8.2 *Error due to competing processes*)
- Any combination of the options above

7.11.8 How to migrate equipments?

Problem

There is no standard migration object available to migrate equipments. The `DEVICE` migration object supports only the migration of devices with a device category of SAP for Utilities. How can equipments and other PM objects be migrated?

Solution

The `BAPI_EQUI_CREATE` function module creates equipments in the SAP system. You can generate a BAPI migration object using this function module. For more information how to generate BAPI migration objects, see chapter 3.4.2 *Generation of a BAPI Migration Object*.

7.11.9 How to migrate historical service notifications?

Problem

The requirement is the migration of a service notification history. The migration strategy according to the functional requirements is:

- Create a service notification (`BAPI_ALM_NOTIF_CREATE` function module)
- Complete the task (`BAPI_ALM_NOTIF_TASK_COMPLETE` function module)
- Change the user status (`BAPI_ALM_NOTIF_CHANGEUSRSTAT` function module)

Each of these BAPI has to be followed by the `BAPI_ALM_NOTIF_SAVE` function module. How can this be implemented in an own migration object?

Solution

This is a typical challenge of migrating a business process. The resulting data has to be saved in the database first before executing the subsequent process step. Therefore, the process cannot be migrated completely with a single migration object. If a single migration object is used, it is not possible to recover from a failed second or third process step even if the data in the import file is corrected. This is because the service function module must assume that the first process step has to be executed in the first try.

Nevertheless, the obvious solution is to use the three BAPI migration object. The challenge is to incorporate the obligatory execution of the `BAPI_ALM_NOTIF_SAVE` function module after each process step. The following solutions available to solve this problem:

- After the generation of the three BAPI migration objects, an event is implemented at the CALL02 generation point (after the call of the service function module) calling the BAPI_ALM_NOTIF_SAVE function module if the service function module (the respective BAPI) finished successfully.
- Implement own service function modules and generate BAPI migration objects based on these function modules. This is done by copying the BAPIs to an own (custom specific) function module, calling the respective BAPI within this function module, and in addition, the BAPI_ALM_NOTIF_SAVE function module.

The second option does not require the implementation of the event as required in the first option. In either case, the result are three BAPI migration objects that are imported in three subsequent steps. In case the first step (creation of the service notification) fails, the subsequent steps will also fail, because the result of the first step (the service notification number) will be required in the subsequent step. You implement this dependency by applying a *Via KSM* field rule for the field with the number of the service notification. For this, you configure the NOTIF_NO field for the second migration object with the *Via KSM* field rule with the name of the first migration object as the **Top Object**, and the third migration object with the same *Via KSM* field rule but the name of the second migration object as the **Top Object**.

7.11.10 Error IO 151: Error while processing the MASE record?

Problem

During data migration of device info records with the DEVINFOREC migration object, the jobs are cancelled with the error IO 151 *Error while processing the MASE record*. How can this problem be resolved?

Solution

The MASE database table contains the current highest serial number used for each material number. It has to be distinguished between the following cases:

- **The serial number of the device info records is given**
The serial number is passed to the application in the automation data. The application verifies whether the given serial number is higher than the serial number stored in MASE database table. If yes, the highest serial number in MASE database table is updated. Potential blockages can be resolved by updating the relevant serial numbers with either the highest possible serial number to be migrated or the highest serial number (999....999). After data migration, the highest serial number has to be determined and adjusted in each record of the MASE database table.
- **No serial for the device info record is given**
The application determines a new unique serial number by accessing the MASE database table, retrieving the highest serial number, increasing the serial number by one, and saving the updated record in the database table. A database lock (ABAP OpenSQL statement `SELECT SINGLE FOR UPDATE`) protects this operation to synchronize parallel accesses and avoid the use of duplicate serial numbers. The application uses the MASE database table like a number range object. This mechanism works fine when serializing device info records online. However, during data migration the retrieval of a new serial number causes a blockage when many jobs are running in parallel with the attempt to create device information records for the same material number. In rare cases, the jobs dump with the error IO 151 *Error while processing the MASE record* or due to a database timeout.

A simple solution is to use an own number range object to determine a new unique serial number in the load program, instead of leaving this determination to the application. In such a way, the performance blockage can be avoided while accessing the MASE database table. The implementation steps are as follows:

1. Define a new ZSERIAL_NR number range object. Alternatively, you can use an existing number range object (for example, EQJUIP_NR). The number range object should be buffered in memory to allow a contention-free retrieval of a unique serial number.
2. Define a ZSERIALNEW fixed value object of the *Code* type with the ABAP code as shown in Figure 7-17.

```

CALL FUNCTION 'NUMBER_GET_NEXT'
  EXPORTING
    NR_RANGE_NR = '01'
    OBJECT = 'ZSERIALNEW'
  IMPORTING
    NUMBER = $$$
    QUANTITY = NUM_CREATED
    RETURNCODE = ERR_CREATED
  EXCEPTIONS
    INTERVAL_NOT_FOUND = 1
    NUMBER_RANGE_NOT_FOUND = 2
    OBJECT_NOT_FOUND = 3
    QUANTITY_IS_0 = 4
    QUANTITY_IS_NOT_1 = 5
    INTERVAL_OVERFLOW = 6
    OTHERS = 7.

```

Figure 7-17: Event to Populate the Fields for the Serial Number

3. Allocate the newly created ZSERIALNEW fixed value object to the INT-GERAET field.
4. Define a ZSERIALGET fixed value object of the type Field and define EMG_DEVMOD_INT-GERAET as the fixed value content.
5. Allocate the newly created ZSERIALGET fixed value object to the INT-GERAET, DEV-GERAET, DEV_FLAG-GERAET, REG-GERAET and REG_FLAG-GERAET fields.
6. Change the X_EXTNUM importing parameter of the ISU_S_SMALL_DEVICE_CREATE service function module from space to X.
7. Allow automatic number allocation in setting the *DevInfoRec.No.Allctn* parameter (technical field name is TE270EGERR-NUMCR_ALLOWED) to X. Figure 7-18 shows the IMG configuration path.

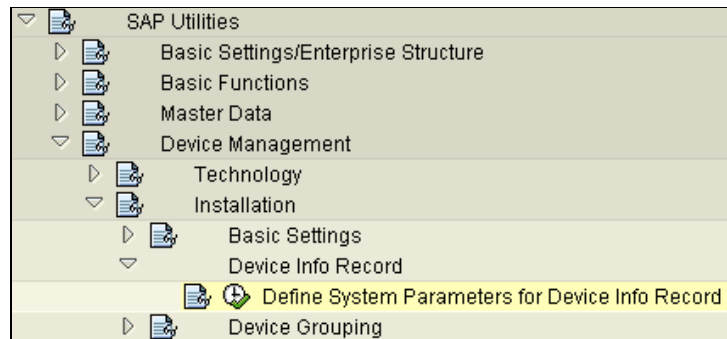


Figure 7-18: IMG Configuration Path

7.11.11 How to use mass processing for BBP_MULT migration object?

Problem

There is a X_MASS parameter in the interface of the ISU_S_BBP_DI1 service function module of the BBP_MULT migration object (budget-billing plan). Is it possible to use this parameter to improve the performance?

Solution

It is possible to use it. For this, two new events for the load program have to be developed. The FKK_CREATE_DOC_MASS_START function module has to be called in an event at the

START generation point and the `FKK_CREATE_DOC_MASS_STOP` function module in an event at the END generation point of the load program of the `BBP_MULT` migration object. It is important to allocate enough number range intervals to the defined document type for budget-billing plans. Otherwise, the number of parallel jobs will be restricted to one job.

7.11.12 How to use mass processing for SECURITY migration object?

Problem

The jobs block each other when retrieving a new number for the request from the document to be created, which is an FI-CA document. Is it possible to use mass processing for the migration of security deposits with the SECURITY migration object?

Solution

To allow mass processing, two events have to be developed in the load program of the SECURITY migration object. The `FKK_CREATE_DOC_MASS_START` function module has to be called in a new event at the START generation point and the `FKK_CREATE_DOC_MASS_STOP` function module in a new event at the END generation point of the load program of the `BBP_MULT` migration object. It is important to allocate enough number range intervals to the defined document type for the request documents. Otherwise, the number of parallel jobs will be restricted to one job.

7.11.13 How to migrate long texts of functional locations?

Problem

You need to migrate long texts (not notes) of function locations, for example, of connection objects and device locations. How can functional location long texts be migrated?

Solution

There is neither a standard migration object nor a BAPI function module available to migrate long texts of functional locations. There exists the `SAVE_TEXT` function module to save a long text. However, you cannot use this function module as a service function module of BAPI migration objects because the definition of its parameter interface does not comply to the design rules of a BAPI (for more information about the design rules, see chapter 3.4.2 *Generation of a BAPI Migration Object*). Therefore, you need to develop an own service function module that calls the `SAVE_TEXT` function module and manage the error handling. Figure 7-19 shows an ABAP code that can be used as a basis for an own development.

```

FUNCTION zzmig_save_long_text.
  *-----
  * Interface local:
  * IMPORTING
  *   REFERENCE(X_HEAD) TYPE THEAD
  * EXPORTING
  *   REFERENCE(RETURN) LIKE BAPI_RET2 STRUCTURE BAPI_RET2
  * TABLES
  *   XT_LINE_TAB STRUCTURE TLINE
  *-----

  * save long text of an technical object
  CLEAR return.
  CALL FUNCTION 'SAVE_TEXT'
    EXPORTING
      client = sy-mandt
      header = x_head
      savemode_direct = 'X'
    TABLES
      lines = xt_line_tab
  EXCEPTIONS
    id = 1
    language = 2
    name = 3
    object = 4

```

```

OTHERS = 5.
IF sy-subrc <> 0.
* return error message
CALL FUNCTION 'BALW_BAPI_RETURN_GET2'
EXPORTING
  type = sy-msgty
  cl = sy-msgid
  number = sy-msgno
  par1 = sy-msgv1
  par2 = sy-msgv2
  par3 = sy-msgv3
  par4 = sy-msgv4
IMPORTING
  return = return.
ENDIF.
ENDFUNCTION.

```

Figure 7-19: Service Function Module to migrate Long Texts of a Function Location

After the generation of the BAPI migration object, change the **Return Field** on the migration object maintenance screen to **AUTO-X_HEAD-TDNAME**. This saves the number of the function location in the KSM. Figure 7-20 shows the field rules of the most relevant fields.

Field Name	Field Rule	Value function location
X_HEAD-TDOBJECT	Fixed Value	IFLOT
X_HEAD-TDNAME	Via KSM	CONNOBJ or DEVLOC
X_HEAD-TDID	Fixed Value	LTXT
X_HEAD-TDSPRAS	Fixed Value	SY-LANGU
X_HEAD-TDFORM	Fixed Value	SYSTEM
X_HEAD-TDLINESIZE	Fixed Value	132
XT_LINE_TAB-TDFORMAT	Fixed Value	*
XT_LINE_TAB-TDLINE	Transfer	Long text

Figure 7-20: Field Rules to migrate Long Texts of a Function Location

7.11.14 How to migrate classification data of function locations?

Problem

SAP delivers the OBJCLASS migration object with field rules to support the migration of classification data of equipments. Which field rules need to be adjusted for a migration of classification data of function locations like connection objects (CONNOBJ migration object) or device locations (DEVLOC migration object)?

Solution

You need to adjust the fields as shown in Figure 7-21.

Field Name	Field Rule	Value function location	Value equipment
HEADER-OBJECT_TYPE	Fixed Value	IFLOT	EQUI
DATA-FIELD	Fixed Value	TPLNR	EQUNR
DATA-VALUE	Via KSM	CONNOBJ or DEVLOC	DEVICE

Figure 7-21: Field Rules to migrate Long Texts of Equipments and Function Location

You can consider the usage of two migration objects instead of a single one if you need to create not only classification data of functional locations but also of equipments.

Alternatively, you can change the field rules of the `HEADER-OBJECT_TYPE` and `DATA-FIELD` field to the *Transfer* field rule. The name of the higher-level migration object (for example, `DEVLOC`, `CONNOBJ`, `DEVICE` or `FUNCLOC`) can be transferred in the import file if the **Gen. Obj. Field** field on the sub screen of the *Via KSM* field rule is marked.

7.11.15 Advanced Metering Infrastructure (AMI) specific data?

Problem

Are there migration objects to migrate AMI specific data?

Solution

The Advanced Metering Infrastructure (AMI) refers to systems that measure, collect and analyze energy usage from advanced devices (smart meters in electricity, gas and water) through various communication media either on request or on a pre-defined schedule. A smart meter generally refers to a type of advanced meter (usually electricity) that tracks consumption in more detail than a conventional. It may also communicate that information via a network back to the local utility for monitoring and billing purposes. A smart grid delivers electricity from suppliers to consumers using digital technology to save energy, reduce costs and increase reliability.

Figure 7-22 shows a list of migration objects where the AMI related fields `AMS` (Advanced Metering System) and `AMCG_CAP_GRP` (Advanced Meter Capability Group AMCG) are available.



You have either to leave both `AMS` and the `AMCG` fields empty or to supply them at the same time.

Migration Object	Description	Structure
DEVICE	Device	EGERH
DEVINFOREC	Device Info Record	DEV
DEVICEMOD	Device Modification	DEV
		DEV_FLAG
INST_MGMT	Device installation / removal / replacement	AUTO_GER

Figure 7-22: AMI related fields

Figure 7-23 shows the migration objects to allocate an Advanced Metering System (AMS) to the regional structure during data migration of a regional structure:

Migration Object	Description	Structure
ADRCITYISU	Utility specific data for city	ADRCITYAMS
ADRSTRTISU	Utility specific data for street	ADRSTRTAMS

Figure 7-23: AMI related fields in Regional Structure

The `METERREAD` migration object has not been enhanced because it is not possible to migrate the transmission status, date and time.



The `NO_AMI_LIMITS` field is a non-migration field and the application ignores it during data migration.

7.11.16 How to migrate formula profiles?

Problem

How can EDM formula profiles be migrated?

Solution

There is neither a standard migration object nor a BAPI function module available to migrate formula profiles. There exists the `ISU_S_FORMINST_CREATE` function module to create formula profiles. However, you cannot use this function module as a service function module of BAPI migration objects because the definition of its parameter interface does not comply to the design rules of a BAPI (for more information about the design rules, see chapter 3.4.2 *Generation of a BAPI Migration Object*). Therefore, you need to develop an own service function module that calls the `ISU_S_FORMINST_CREATE` function module and manage the error handling. Figure 7-24 shows an ABAP code that can be used as a basis for an own development.

```

FUNCTION zzmig_edm_forminst_create .
  *-----
  * Interface local:
  * IMPORTING
  *   VALUE(HEAD) TYPE  EFORMINST_AUTO_HEAD
  * EXPORTING
  *   REFERENCE(EX_INSTANCE_NUMBER) TYPE  E_INSTANCE_NUMBER
  *   REFERENCE(RETURN) LIKE  BAPI_RET2 STRUCTURE  BAPI_RET2
  * TABLES
  *   EFORMINST_PROFILE STRUCTURE  EPROFHEAD_AUTO_HEAD
  *   INPUT_PARAMETER_S STRUCTURE  EFORMINST_PARAM_I_AUTO_S
  *   OUTPUT_PARAMETER_S STRUCTURE  EFORMINST_PARAM_O_AUTO_S
  *-----

  * local data definition
  DATA: lv_auto TYPE  eforminst_auto_s,
        wa_eforminst_profile TYPE  eprofhead_auto_head,
        lv_eforminst_profile TYPE  eforminst_profile_auto,
        wa_input_parameter TYPE  eforminst_param_i_auto_s,
        wa_output_parameter TYPE  eforminst_param_o_auto_s,
        lv_db_update TYPE  regen-db_update.

  * fill header data
  lv_auto-head = head.

  * add input/output parameter
  lv_auto-input_parameter_s[] = input_parameter_s[].
  lv_auto-output_parameter_s[] = output_parameter_s[].

  * add profiles to be created
  LOOP AT eforminst_profile INTO wa_eforminst_profile.
    lv_eforminst_profile-profile_to_create = sy-tabix.
    lv_eforminst_profile-eprofhead_auto_head = wa_eforminst_profile.
    APPEND lv_eforminst_profile TO lv_auto-eforminst_profile.
  ENDLOOP.

  * Check link to profile to create (input parameter)
  LOOP AT input_parameter_s INTO wa_input_parameter.
    IF NOT wa_input_parameter-profile_to_create IS INITIAL.
      READ TABLE lv_auto-eforminst_profile
        INDEX wa_input_parameter-profile_to_create
        TRANSPORTING NO FIELDS.
      IF sy-subrc NE 0.
        * profile not found
        CALL FUNCTION 'BALW_BAPI_RETURN_GET2'
          EXPORTING
            type      = 'E'
            cl        = 'EPRO'
            number    = '423'
          IMPORTING
            return    = return.
      ENDIF.
    ENDIF.
  ENDLOOP.

```

```

        IF 1 = 2. MESSAGE e423(epro). ENDIF.                "#EC BOOL_OK
        EXIT.
    ENDF.
ENDIF.
ENDLOOP.

* Check link to profile to create (output parameter)
LOOP AT output_parameter_s INTO wa_output_parameter.
    IF NOT wa_output_parameter-profile_to_create IS INITIAL.
        READ TABLE lv_auto-eforminst_profile
            INDEX wa_output_parameter-profile_to_create
            TRANSPORTING NO FIELDS.
        IF sy-subrc NE 0.
*           profile not found
            CALL FUNCTION 'BALW_BAPI_RETURN_GET2'
                EXPORTING
                    type = 'E'
                    cl   = 'EPRO'
                    number = '423'
                IMPORTING
                    return = return.
            IF 1 = 2. MESSAGE e423(epro). ENDIF.                "#EC BOOL_OK
            EXIT.
        ENDF.
    ENDF.
ENDLOOP.

* create formula instance
CALL FUNCTION 'ISU_S_FORMINST_CREATE'
    IMPORTING
        ex_instance_number = ex_instance_number
        ex_db_update       = lv_db_update
    CHANGING
        ch_auto            = lv_auto
    EXCEPTIONS
        OTHERS             = 1.
IF sy-subrc <> 0.
* fill RETURN with error messages
    CALL FUNCTION 'BALW_BAPI_RETURN_GET2'
        EXPORTING
            type = sy-msgty
            cl   = sy-msgid
            number = sy-msgno
            par1 = sy-msgv1
            par2 = sy-msgv2
            par3 = sy-msgv3
            par4 = sy-msgv4
        IMPORTING
            return = return.
    ELSEIF lv_db_update IS INITIAL.
        CALL FUNCTION 'BALW_BAPI_RETURN_GET2'
            EXPORTING
                type = 'E'
                cl   = 'EM'
                number = '105'
            IMPORTING
                return = return.
        ENDF.
    ENDFUNCTION.

```

Figure 7-24: Service Function Module to migrate Formula Profiles

After the generation of the BAPI migration object, change the **Return Field** on the migration object maintenance screen to AUTO-EXPORT_FIELDS-EX_INSTANCE_NUMBER. This saves the formula calculation number in the KSM.



The newly created migration object allows the creation of a formula profile and its input and output parameters (INPUT_PARAMETER_S structure) and output (OUTPUT_PARAMETER_S structure). If you want required input and output profiles to be created simultaneously, you must allocate a sequence number (PROFILE_TO_CREATE field)

to the related input and output parameter. The sequence number must correspond to the number of the transferred `EFORMINST_PROFILE` structure in the import file.

7.12 FAQ Miscellaneous

7.12.1 What are the correct dates for migrating data?

Problem

What are the correct dates to use when migrating master and transaction data?

Solution

This depends on the migration strategy to be implemented. Some migration objects contain a date that has to be exact and unique for each business partner, and some have dates that can be the same for all objects. Migration objects that can be considered as master data, can have the same date for all migration objects, but when they represent a transaction the extraction program needs to find the exact relevant date.

One example where you can use the same date is the validity date for business partner. The important thing for these dates is that no transaction can be performed on a date earlier than this creation date. It is common to set these creation dates to 01/01/1900, for example. This should work for all normal purposes.

The other set of dates is related to transaction data being migrated. Some important scenarios are the posting date of a financial data (for example, open item or payment) and the installation date of a device and the move-in date of a contract (for example, in SAP for Utilities). For such scenarios, the extractors need to build a logic to evaluate these dates for each object and business partner. When the migration object for installing the meter is built, it requires the date and the meter reading relevant for this date.

This document explains for the most important scenarios, which dates should be used or are most commonly used. For a list of migration strategies, see also chapter 6 *Migration Strategies (SAP for Utilities)*.

7.12.2 Runtime error with condition OBJECT_NOT_FOUND?

Problem

The load report terminates with the `RAISE_EXCEPTION` runtime error with the `OBJECT_NOT_FOUND` exception condition. The error analysis of the short dump shows a statement in the `SAPLSLG` program raised the `OBJECT_NOT_FOUND` exception condition.

Solution

The most likely cause of the program termination is that a high number of problems have been identified during the data import. Each error leads to at least two entries in the error (application) log. The application log can only hold a maximum of 999.999 messages. Usually, one of the following causes the problem:

- A wrong load order of import files or replaced import files leads to a rejection of all data objects due to a missing higher-level object while processing the *Via KSM* field rule. The error message EM 102 *The higher-level object for x, y, does not exist*, is raised for many data objects.
- Almost all data objects have already been migrated but the data import was not started in *restart* mode. The error message EM 101 *Legacy system key x has already been migrated* is raised for many data objects.
- A customer-specific event in the load report is raising an error message in the wrong way (for more information about how to process an error message correctly, see chapter 4.3.4 *Error Handling in Custom Code*).

The application log is not accessible when a program cancels or is cancelled; this impedes the analysis of the problem. In addition to writing the error messages in the application log, it is possible to have them written to the job log that is available after manual termination of the

job. You can enable this function by marking the *Message in Job Log* field in the user parameter on the *Monitoring* sub screen.



This function leads to a poor performance of the load program. Therefore, you should enable this function only until the problem is fixed.

7.12.3 How to send e-mails and SMS during data migration?

Problem

The data migration team wants you to keep them informed about the progress of the execution of the data migration by email or SMS. How can you do this?

Solution

SAPconnect provides a standard interface for external communication, which supports telecommunication services, such as, FAX, text messages (pager or SMS), Internet mail, and X.400. It can also send messages to printers and between different SAP Systems. It enables external communication components to be connected to the SAP System.

SAPconnect provides a direct connection to the Internet through the Simple Mail Transfer Protocol (SMTP) plug-in of the SAP Web Application Server. This enables you to send and receive emails, faxes, and text messages (pager or SMS) directly, without having to use external communication systems. For more information, see **SAP Exchange Connector** and **SMTP Configuration Guide**.

After the configuration of SAPconnect, a customer-specific event in the Distributed Import of the IS Migration Workbench can be developed to send emails and SMS (for more information, see chapter 9.5 *Function Module to Send a Status Email During Data Import*).

7.12.4 How to debug a load program

Problem

The data migration team wants debug the data load. How do you put the load program into debug mode?

Solution

There are several options available:

- Choose the *Dialog Work Process* radio button on the data import screen and then *Utilities* → *Import /debug mode*). The load program stops and changes into debug mode at the `MAC_BREAK` macro statement just before calling the service function module. This macro (section of source code in ABAP) executes a break point, if the `P_DEBUG` global variable in the load program is set to X. The `P_DEBUG` variable can be changed in debug mode.
- Enter the `DEBUG` command in the command line when displaying the data import screen. After you have chosen the *Data Import* button, the load application changes into debug mode just before it submits the load program.
- Implement an event of *Code* type in the load program with the `BREAK-POINT` ABAP statement. For more information about how to implement an event, see chapter 4.3 *Custom Code in a Load Program*. The load program is put into debug mode when it reaches the `BREAK-POINT` statement.

7.12.5 How to create a migration object to execute an ABAP program?

Problem

The data migration team wants you to generate a migration object based on an ABAP program (report). How can you do this?

Solution

You need to develop a service function module that you use for the generation of a BAPI migration object (for more information how to generate BAPI migration objects, see chapter 3.4.2 *Generation of a BAPI Migration Object*). Figure 9-10 shows an ABAP code that can be used as a basis for an own development. The function module submits a report based either on a given variant or on a list of selection screen parameters. You must transfer the required information in an import file.

7.12.6 How to Execute Programs in a Group Import?

Problem

The data migration team wants you to execute an own report as part of a group import in order to automate the data migration process. How can you do this?

Solution

The group import allows the execution of a sequence of import runs based on the distributed import. Therefore, you need to define a profile of a distributed import based on an own migration object that executes your report. For more information how to create such a migration object, see chapter 7.12.5 *How to create a migration object to execute an ABAP program?*

8 Migration from SAP System to SAP System

You must distinguish between the two fundamental approaches of a data migration based on business objects and a data migration based on database tables. Each project has to analyze the positives and negatives of both approaches and their feasibility regarding costs and effort.

For a data migration based on business objects, it is important to understand the following:

- The IS Migration Workbench is a tool designed to load data into an SAP system based on business objects and business processes.
- SAP does not deliver any extraction tool for an SAP system for migration purposes based on business objects. This means that a project must develop own extraction and transformation programs, or use software from SAP partners for the extraction.

Such an approach is typically implemented when the implementations of the present and new SAP system differ too much, specifically concerning release and Customizing of the solutions.

For a data migration based on database tables, the SAP AGS SLO team supports customers world-wide to implement business-driven change or harmonization requirements in their live SAP environments. They do this by providing the required technologies, tools, and services. In a joint effort with the Global SLO (System Landscape Optimization) Hub, SAP Solution Support, and selected SAP partners, the AGS SLO team provides the following:

Tools

- SAP Test Data Migration Server
Helps to minimize hardware and system administration costs by automating the creation of consistent non-productive systems
- Client Migration Server (C-Room):
Tool available as part of an SAP Consulting service for Client Transfer projects to analyze, migrate, or upgrade clients from one SAP system to another SAP system

Services

- System Landscape Optimization services
Available for system consolidation, data harmonization and organizational changes such as such as controlling area, company code, plants, material number, profit centers, *PSP* structures, cost centers, FI accounts
- SAP General Ledger Migration
Standardized service to switch to new general ledger
- Currency and supplementary SLO services
For example, Euro conversion or archive conversion or coding scan

For more information about the available services, see the homepage of SAP AGS SLO on <http://service.sap.com/slo>.

9 Appendix with ABAP Code Snippets

9.1 Service Function Module for EDM RTP Interface

Chapter 6.11 *Energy Data Management (EDM)* describes the need for a service function module to migrate RTP interfaces for EDM. The code snippet in Figure 9-1 can be used as a basis for an own implementation.

```

FUNCTION z_migration_rtp_interface.
*-----
* Local Interface:
* IMPORTING
*   REFERENCE(X_ANLAGE) TYPE ANLAGE
*   REFERENCE(X_DATE) TYPE DATUM
* EXPORTING
*   REFERENCE(RETURN) TYPE BAPIRET2
* TABLES
*   X_AUTO_PROFILE STRUCTURE DERTPIFACEINSTPAR OPTIONAL
*   X_AUTO_ACTIVATION STRUCTURE DERTPIFACEINST OPTIONAL
* EXCEPTIONS
*   GENERAL_FAULT
*   NOT_FOUND
*-----

* Local data definition
DATA: lt_auto_profile TYPE tdertpifaceinstpar,
      lt_auto_activation TYPE tdertpifaceinst,
      y_auto TYPE eedm_obj_hisrelate.

* copy tables into local variables defined as table types
lt_auto_profile = x_auto_profile[].
lt_auto_activation = x_auto_activation[].

* call the service function module
CALL FUNCTION 'ISU_EEDM_S_AUTO_LINK'
  EXPORTING
    x_anlage           = x_anlage
    x_auto_profile     = lt_auto_profile
    x_auto_activation  = lt_auto_activation
    x_date             = x_date
  IMPORTING
    y_auto             = y_auto
  EXCEPTIONS
    general_fault      = 1
    not_found          = 2
    OTHERS              = 3.

* error handling
IF sy-subrc <> 0.
  CALL FUNCTION 'BALW_BAPI_RETURN_GET2'
    EXPORTING
      type = 'E'
      cl   = sy-msgid
      number = sy-msgno
      par1  = sy-msgv1
      par2  = sy-msgv2
      par3  = sy-msgv3
      par4  = sy-msgv4
    IMPORTING
      return = return.
ENDIF.
ENDFUNCTION.

```

Figure 9-1: Service Function Module for EDM RTP Interface

9.2 Direct Input Modules for Contract Account (BDT)

Chapter 4.4.6 *Enhancement Fields for Contract Account* describes how you must use the BDT tool to implement enhancement fields for contract accounts. The code snippet in Figure 9-2 can be used as a basis for an own implementation to transfer and save the contents of enhancement fields in the database during direct input thus in data migration.

```

FUNCTION ZCI_FI_CA_EVENT_DINP2.
*-----
*"" Lokale Schnittstelle:
*   IMPORTING
*       VALUE(I_DATA) LIKE BUSODIAT1 STRUCTURE BUSODIAT1
*-----

DATA: L_FKKVK_DI_HDRINIT LIKE FKKVK_DI_HDRINIT,
      L_FKKVK_S_DI_OLD LIKE FKKVK_S_DI,
      L_FKKVK_S_DI_TRANS LIKE FKKVK_S_DI,
      L_FKKVKP_S_DI_OLD LIKE FKKVKP_S_DI,
      L_FKKVKP_S_DI_TRANS LIKE FKKVKP_S_DI.

*----- process common data part -----
CASE I_DATA-TBNAM.
  WHEN ' FKKVK_S_DI '.

* ---- Nothing to do: Because IS-U has no fields with common data

* - partnerspecific data
  WHEN ' FKKVKP_S_DI '.

* ---- Store actual partner data in local mem
  IF NOT ACTUAL_BP IS INITIAL.
    PERFORM STORE_ACTUAL_DATA_IN_IT_FKKVKP USING ACTUAL_BP.
  ENDIF.

* ---- get data from parameter
  L_FKKVKP_S_DI_TRANS = I_DATA-DATA.

* ---- first try to get the current partner from local memory
  READ TABLE IT_FKKVKP WITH KEY GPART = L_FKKVKP_S_DI_TRANS-PARTNER.
  IF SY-SUBRC = 0.
    FKKVKP = IT_FKKVKP.

* ----- combine old and new given data
  MOVE-CORRESPONDING FKKVKP TO L_FKKVKP_S_DI_OLD.
  ELSE.
    CLEAR L_FKKVKP_S_DI_OLD.
  ENDIF.

  CALL FUNCTION 'BUS_DI_DATA_COMBINE'
    EXPORTING
      I_STRUCTURE          = ' FKKVKP_S_DI '
      I_DATA_OLD           = L_FKKVKP_S_DI_OLD
      I_DATA_TRANSFERRED   = L_FKKVKP_S_DI_TRANS
    IMPORTING
      E_DATA_NEW           = FKKVKP_S_DI.

* ---- store partner data in dynpro fields for checking in PAI
  MOVE-CORRESPONDING FKKVKP_S_DI TO IT_FKKVKP.
  IT_FKKVKP-GPART = L_FKKVKP_S_DI_TRANS-PARTNER.
  MOVE-CORRESPONDING IT_FKKVKP TO CI_FKKVKP.

* ---- set actual BP
  ACTUAL_BP = L_FKKVKP_S_DI_TRANS-PARTNER.

ENDCASE.

ENDFUNCTION.

FUNCTION ZCI_FI_CA_EVENT_DSAVB.
*-----
*"" Lokale Schnittstelle:
*   IMPORTING
*       VALUE(I_DATA) LIKE BUSODIAT1 STRUCTURE BUSODIAT1

```

```

** -----
DATA: SUBNAME LIKE DD02D-TABNAME VALUE 'CI_FKKVKP',
      L_FKKVKP LIKE FKKVKP.

* Store current dypn-values in application-local table
PERFORM STORE_ACTUAL_DATA_IN_IT_FKKVKP USING ACTUAL_BP.

* The Owner-Application collect the data
CALL FUNCTION 'VKK_FICA_FKKVKP_COLLECT'
  EXPORTING
    I_SUBNAME = SUBNAME
  TABLES
    PT_FKKVKP = IT_FKKVKP.

ENDFUNCTION.
FUNCTION ZCI_FICA_EVENT_ISDST.
** -----
** *** Lokale Schnittstelle:
** -----

* initialize actual buisness partner
CLEAR ACTUAL_BP.

* Get data from the main Application Vkk (table-owner)
CALL FUNCTION 'VKK_FICA_FKKVKP_GET'
  TABLES
    T_FKKVKP = IT_FKKVKP.

CALL FUNCTION 'VKK_FICA_FKKVK_GET'
  IMPORTING
    E_FKKVK = FKKVK_OLD.

APPLIKAT = FKKVK_OLD-APPLK.
IF APPLIKAT IS INITIAL.
  CALL FUNCTION 'FKK_GET_APPLIKATION'
    IMPORTING
      E_APPLK = APPLIKAT
    EXCEPTIONS
      NO_APPL_SELECTED = 1
      OTHERS = 2.
ENDIF.

IF NOT FKKVK_OLD-VKTYP IS INITIAL AND
   NOT FKKVK_OLD-APPLK IS INITIAL.
  CALL FUNCTION 'FKK_TFK002A_READ'
    EXPORTING
      VKTYP = FKKVK_OLD-VKTYP
      APPLK = FKKVK_OLD-APPLK
    IMPORTING
      E_TFK002A = TFK002A
    EXCEPTIONS
      NOT_FOUND = 1
      SYSTEM_ERROR = 2
      OTHERS = 3.
  IF SY-SUBRC NE 0.
    CALL FUNCTION 'BUS_MESSAGE_STORE'
      EXPORTING
        ARGBB = SY-MSGID
        MSGTY = VK_MSG_ERROR
        TXTNR = SY-MSGNO
        MSGV1 = SY-MSGV1
        MSGV2 = SY-MSGV2
        MSGV3 = SY-MSGV3
        MSGV4 = SY-MSGV4.
  ENDIF.
ENDIF.

* Store old data in local mem for comparing old and new values later
CHECK VK_STAT-AKTYP NE VK_AKTYP_CREATE.
IT_FKKVKP_OLD[] = IT_FKKVKP[].

ENDFUNCTION.
FUNCTION ZCI_FICA_EVENT_XCHNG.
** -----

```

```

**" Lokale Schnittstelle:
**      EXPORTING
**          VALUE(E_XCHNG) LIKE BUS000FLDS-XCHNG
**
-----
CALL FUNCTION 'VKK_FI_CA_ACTUAL_BP'
  IMPORTING
    E_GPART = ACTUAL_BP
  EXCEPTIONS
    OTHERS   = 1.

* First store actual data in it_fkkvvp
PERFORM STORE_ACTUAL_DATA_IN_ITFKVKVP USING ACTUAL_BP.

* First sort table
SORT IT_FKKVKP BY GPART.
SORT IT_FKKVKP_OLD BY GPART.

* check if something changed
IF IT_FKKVKP[] <> IT_FKKVKP_OLD[].
  E_XCHNG = TRUE.
ELSE.
  E_XCHNG = FALSE.
ENDIF.

ENDFUNCTION.

```

Figure 9-2: Example of an Implementation of the BDT for Contract Accounts

9.3 Service function Module to Migrate a Custom Table

Chapter 5.3 *Migrating Custom Tables Using an Own Migration Object* describes how to implement an own migration object for the migration of custom tables. The code snippet in Figure 9-3 can be used as a basis for an own implementation. For example, you can add more validations of the transferred data.

```

FUNCTION z_mig_migration_ztable.
**" -----
**" Local Interface:
**      IMPORTING
**          REFERENCE(X_UPDATE) TYPE EMG_UPDONLINE DEFAULT SPACE
**      EXPORTING
**          REFERENCE(RETURN) TYPE BAPI_RETURN
**      TABLES
**          XT_ZTABLE_TAB STRUCTURE ZTABLE
**" -----

* Local variables
DATA: lv_ztable TYPE ztable,
      lv_error TYPE c.

* Initialization
CLEAR lv_error.

* check if rows already exist
IF NOT xt_ztable_tab[] IS INITIAL.
  LOOP AT xt_ztable_tab INTO lv_ztable.
    SELECT SINGLE * FROM ztable INTO lv_ztable
      WHERE partner = lv_ztable-partner
      AND lfdnr = lv_ztable-lfdnr.

* error handling
IF sy-subrc = 0.
  CALL FUNCTION 'BALW_BAPI_RETURN_GET'
    EXPORTING
      type      = 'E'
      cl        = 'EM'
      number    = '000'
      par1      = 'Error updating ZTABLE for'
      par2      = lv_ztable-partner
      par3      = ' '
      par4      = ' '

```

```

IMPORTING
    bapi_return = return.
lv_error = 'X'.
EXIT.
ENDIF.
ENDLOOP.
ENDIF.

* update database table
IF lv_error <> 'X'.

    IF x_update EQ 'X'.
*       update database table on commit
        APPEND LINES OF xt_ztable_tab TO gt_ztable_data.
        PERFORM update_ztable_data ON COMMIT.
        PERFORM refresh_ztable_data ON ROLLBACK.
    ELSE.
*       direct database table update
        INSERT ztable FROM TABLE xt_ztable_tab.
    ENDIF.

ENDIF.

ENDFUNCTION.

*&-----*
*&       Form   update_ztable_data
*&-----*
FORM update_ztable_data.
    INSERT ztable FROM TABLE gt_ztable_data.
    REFRESH gt_ztable_data.
ENDFORM.                "update_ztable_data

*&-----*
*&       Form   refresh_ztable_data
*&-----*
FORM refresh_ztable_data.
    REFRESH gt_ztable_data.
ENDFORM.                "refresh_ztable_data
ENDFUNCTION.

Top-Include of the function group:
FUNCTION-POOL zmigmigration.

DATA gt_ztable_data TYPE ztable OCCURS 0.

```

Figure 9-3: Service Function Module to Migrate a Custom Table

You can easily enhance the code snippet above if you need to update more than one custom table or you want to pass information to a header table in which you refer to the second table. The code snippet in Figure 9-3 shows the `Z_MIG_MIGRATION_ZTABLE_HEADER` function module that can be used as a basis for an own implementation. The primary key of the `ZTABLE_HEADER` database table consists out of the `MANDT` field and the `PARTNER` field.

```

FUNCTION zmigmigration_ztable.
*-----*
*"" Local Interface:
*"" IMPORTING
*""     REFERENCE(X_UPDATE) TYPE EMG_UPDONLINE DEFAULT SPACE
*""     REFERENCE(X_ZTABLE_HEADER) TYPE ZTABLE_HEADER
*"" EXPORTING
*""     REFERENCE(RETURN) TYPE BAPI_RETURN
*"" TABLES
*""     XT_ZTABLE_TAB STRUCTURE ZTABLE
*-----*

* local variables
DATA: lv_ztable_header TYPE ztable_header,
      lv_error TYPE c.

```

```

* Initialisation
CLEAR lv_error.

* check if rows already exist
SELECT SINGLE * FROM ztable_header INTO lv_ztable_header
WHERE partner = lv_ztable_header-partner.

* error handling
IF sy-subrc = 0.
  CALL FUNCTION 'BALW_BAPI_RETURN_GET'
    EXPORTING
      type      = 'E'
      cl        = 'EM'
      number    = '000'
      par1      = 'Error updating ZTABLE for'
      par2      = lv_ztable_header-partner
      par3      = ' '
      par4      = ' '
    IMPORTING
      bapi_return = return.
  lv_error = 'X'.
  EXIT.
ENDIF.

* update database table
IF lv_error <> 'X'.

  IF x_update EQ 'X'.
    * update database table on commit
    APPEND x_ztable_header TO gt_ztable_header_data.
    APPEND LINES OF xt_ztable_tab TO gt_ztable_data.
    PERFORM update_ztable_data ON COMMIT.
    PERFORM refresh_ztable_data ON ROLLBACK.
  ELSE.
    * direct database table update
    INSERT ztable_header FROM x_ztable_header.
    INSERT ztable FROM TABLE xt_ztable_tab.
  ENDIF.
ENDIF.

ENDFUNCTION.

*&-----*
*&      Form  update_ztable_data
*&-----*
FORM update_ztable_data.
  INSERT ztable_header FROM x_ztable_header.
  INSERT ztable FROM TABLE gt_ztable_data.
  REFRESH gt_ztable_header_data.
  REFRESH gt_ztable_data.
ENDFORM.                "update_ztable_data

*&-----*
*&      Form  refresh_ztable_data
*&-----*
FORM refresh_ztable_data.
  REFRESH gt_ztable_header_data.
  REFRESH gt_ztable_data.
ENDFORM.                "refresh_ztable_data
ENDFUNCTION.

Top-Include of the function group:
FUNCTION-POOL zmig_migration.
DATA gt_ztable_header_data TYPE ztable_header OCCURS 0.
DATA gt_ztable_data TYPE ztable OCCURS 0.

```

Figure 9-4: Service Function Module to Migrate a Custom Table with Header Table

9.4 Function Module to Determine Reconciliation Keys for a Reuse

Chapter 7.7.2 *Is it possible to reuse reconciliation key* describes how to reuse reconciliation keys during data migration of financial documents. The code snippet in Figure 9-5 can be used as a basis for an own implementation. It is based on the `ISU_M_FIKEY_DETERMINE` function module and a `ZMIG_FIKEY` custom database table, which is defined by the `MANDT` (type `MANDT`), `OBJECT` (type `EMG_OBJECT`) (name of migration object), and `FIKEY` (type `FIKEY_KK`) fields. Define the `G_FIKEY` global variable in the top-include of the function group.

The data migration team has to save the reconciliation keys to be used in the database table before a data migration. The number of saved reconciliation keys limits the number of parallel jobs because each job requires its own reconciliation keys.

```

FUNCTION z_isu_m_fikey_determine_reuse.
** -----
** Local Interface:
** IMPORTING
**   VALUE(X_OBJECT) TYPE EMG_OBJECT
** EXPORTING
**   VALUE(Y_FIKEY) TYPE FIKEY_KK
** EXCEPTIONS
**   NO_TEMPFIKEY
** -----

* Local variable definition
DATA: lt_zmig_fikey TYPE zmig_fikey OCCURS 0,
      wa_zmig_fikey TYPE zmig_fikey,
      lv_guid TYPE guid_32,
      lv_pointer LIKE sy-index.

* check if fikey has already been determined
IF g_fikey IS INITIAL.

* initialization
CLEAR y_fikey.

* try until a free recon key can be blocked
* risk of an endless loop!
DO.

* load preconfigured reconciliation keys from database
SELECT * FROM zmig_fikey INTO TABLE lt_zmig_fikey
WHERE object = x_object.

* raise error message if no reconciliation key has been defined
IF sy-subrc <> 0.
  MESSAGE a205(em) WITH 'ZMIG_FIKEY' space RAISING no_tempfikey.
ENDIF.

* determine guid for random tries
call function 'GUID_CREATE'
  importing
    ev_guid_32 = lv_guid.
TRANSLATE lv_guid(6) USING 'A9B8C7D6E5F4'.
CLEAR lv_pointer.

DO.

* Delete the recon key of the last try
IF lv_pointer > 0.
  DELETE lt_zmig_fikey INDEX lv_pointer.
ENDIF.

* Leave loop if all recon key have been tested
DESCRIBE TABLE lt_zmig_fikey LINES sy-tfill.
IF sy-tfill = 0.
  EXIT.
ENDIF.

```

```

*      Calculate new index
lv_pointer = lv_guid(6) MOD sy-tfill + 1.
READ TABLE lt_zmi_g_fi_key INTO wa_zmi_g_fi_key INDEX lv_pointer.

*      Try to block reconciliation key
CALL FUNCTION 'ENQUEUE_EFKKFKEY'
  EXPORTING
    x_fi_key      = wa_zmi_g_fi_key-fi_key
    _scope        = '1'
  EXCEPTIONS
    foreign_lock  = 01
    system_failure = 02.

*      if reconciliation key could be blocked leave loop
IF sy-subrc = 0.
  y_fi_key = ws_zmi_g_fi_key-fi_key.
  EXIT.
ENDIF.

ENDDO.

*      if reconciliation key could be blocked leave loop
IF NOT y_fi_key IS INITIAL.
  EXIT.
ENDIF.

ENDDO.

*      save determined recon key in global memory
g_fi_key = y_fi_key.

ELSE.

*      return what we have already
y_fi_key = g_fi_key.

ENDIF.

ENDFUNCTION.

```

Figure 9-5: Function Module to Determine Reconciliation Keys for a Reuse

9.5 Function Module to Send a Status Email During Data Import

Chapter 2.7.5 *User-Exit in Master Job* describes how to implement a custom-specific function module during the execution of a distributed import. The code snippet in Figure 9-6 is an example and it can be used as a basis for an own implementation. It is a copy of the ISU_M_SAMPLE_DISTR_IMPORT standard function module and calls the SO_NEW_DOCUMENT_ATT_SEND_API1 standard function module to keep the migration team informed about the status of the distributed import by email. Of course, the example can be enhanced easily, for example, by sending more sophisticated information or using a recipient list in a custom table.

```

FUNCTION z_isu_m_sample_distr_import.
** -----
** "" Local Interface:
** ""
** "" IMPORTING
** ""   VALUE(X_STATUS) TYPE   EMG_STATUS OPTIONAL
** ""   VALUE(X_RUNGROUP) TYPE  EMG_RUNGROUP OPTIONAL
** ""   VALUE(X_RUNGROUP_SEQ_NUMBER) TYPE  EMG_SEQ_NUMBER OPTIONAL
** ""   VALUE(X_TEMRUN) TYPE    TEMRUN OPTIONAL
** ""   VALUE(X_TEMRUNMASTER) TYPE  TEMRUNMASTER OPTIONAL
** ""   VALUE(X_TEMSTATISTIK) TYPE  TEMSTATISTIK OPTIONAL
** ""
** "" EXPORTING
** ""   VALUE(Y_STATUS) TYPE   EMG_STATUS
** ""
** "" TABLES
** ""   T_RUNDIST STRUCTURE  TEMRUNDIST OPTIONAL
** ""   T_RUNJOBS STRUCTURE  TEMRUNJOBS OPTIONAL
** "" -----

```

```

* local data definition
DATA: lt_objpack TYPE sopcklst1 OCCURS 0,
      wa_objpack TYPE sopcklst1.
DATA: lt_objbin TYPE solisti1 OCCURS 0.
DATA: lt_objtxt TYPE solisti1 OCCURS 0,
      wa_objtxt TYPE solisti1.
DATA: lt_reclist TYPE somlreci1 OCCURS 0,
      wa_reclist TYPE somlreci1.
DATA: lv_doc_data TYPE sodocchgi1.
DATA: lv_tab_lines TYPE sy-tabix.
DATA: lv_line(80) TYPE c.

* macro definition
DEFINE mac_add_line.
  wa_objtxt-line = &1.
  append wa_objtxt to lt_objtxt.
END-OF-DEFINITION .
DEFINE mac_add_recipient.
  wa_reclist-receiver = &1.
  wa_reclist-rec_type = 'U'.
  append wa_reclist to lt_reclist.
END-OF-DEFINITION .

* check status of data load
CHECK x_status NE 'RUN'.

* create subject of email
lv_doc_data-obj_name = 'Data Migration'.
CONCATENATE 'Status Data Migration ' x_temrun-object x_status
  INTO lv_doc_data-obj_descr SEPARATED BY space.

* create email body

" write system date & time
WRITE sy-datum TO lv_line.
WRITE sy-uzeit TO lv_line+11.
mac_add_line lv_line.

" write statistic information
WRITE: x_temstatistik-cnt_good TO lv_line(10).
CONCATENATE 'Count Good:' lv_line(10)
  INTO lv_line SEPARATED BY space.
mac_add_line lv_line.
WRITE: x_temstatistik-cnt_bad TO lv_line(10).
CONCATENATE 'Count Bad:' lv_line(10)
  INTO lv_line SEPARATED BY space.
mac_add_line lv_line.

* calculate size if email body
DESCRIBE TABLE lt_objtxt LINES lv_tab_lines.
READ TABLE lt_objtxt INTO wa_objtxt INDEX lv_tab_lines.
lv_doc_data-doc_size = ( lv_tab_lines - 1 ) * 255 +
  STRLEN( wa_objtxt-line ).

* create entry for compressed document
CLEAR wa_objpack-transf_bin.
wa_objpack-head_start = 1.
wa_objpack-head_num = 0.
wa_objpack-body_start = 1.
wa_objpack-body_num = lv_tab_lines.
wa_objpack-doc_type = 'RAW'.
APPEND wa_objpack TO lt_objpack.

* recipient list
mac_add_recipient 'migration@project.com'.

* send the email
CALL FUNCTION 'SO_NEW_DOCUMENT_ATT_SEND_API1'
  EXPORTING
    document_data      = lv_doc_data
    put_in_outbox      = 'X'
    commit_work        = 'X'
  TABLES
    packing_list       = lt_objpack
    contents_bin       = lt_objbin

```

```

        contents_txt      = lt_objtxt
        recei vers        = lt_reclist
    EXCEPTIONS
        too_many_recei vers    = 1
        document_not_sent      = 2
        operation_no_authori zati on = 4
        OTHERS                  = 99.
ENDFUNCTION.

```

Figure 9-6: Function Module to Send a Status Email during Data Import

9.6 Service Function Modules for Service Notifications

Chapter 7.11.9 *How to migrate historical service notifications* describes how to implement custom-specific service function modules for the migration of historical service notifications. The code snippet in Figure 9-7, Figure 9-8 and Figure 9-9 are examples and can be used as a basis for an own implementation.

You can create the `Z_BAPI_ALM_NOTIF_CREATE` function module by copying the `BAPI_ALM_NOTIF_CREATE` and then deleting all the copied ABAP code. With the copy, the custom-specific function module has exactly the same parameter interface as the BAPI. Now all of the ABAP code is replaced by a call of the `BAPI_ALM_NOTIF_CREATE` function module (passing all the parameter of the interface) followed by the call of the `BAPI_ALM_NOTIF_SAVE` function module. After the generation of the BAPI migration object, change the **Return Field** on the migration object maintenance screen to `AUTO-NOTIFHEADER_EXPORT-NOTIF_NO`. This saves the number of the created service notification in the KSM.

```

FUNCTION z_bapi_alm_noti f_create.
*-----
* " "Interface local:
* " IMPORTING
* "     VALUE(EXTERNAL_NUMBER) LIKE BAPI2080_NOTHDRE-NOTIF_NO OPTIONAL
* "     VALUE(NOTIF_TYPE) LIKE BAPI2080-NOTIF_TYPE
* "     VALUE(NOTIFHEADER) LIKE BAPI2080_NOTHDRI STRUCTURE
* "         BAPI2080_NOTHDRI
* "     VALUE(TASK_DETERMINATION) LIKE BAPIFLAG STRUCTURE BAPIFLAG
* "         DEFAULT SPACE
* "     VALUE(SENDER) LIKE BAPI_SENDER STRUCTURE BAPI_SENDER OPTIONAL
* "     VALUE(ORDERID) LIKE BAPI2080_NOTHDRE-ORDERID OPTIONAL
* "     REFERENCE(I_NR_MAX_NOTES) TYPE INT_1 OPTIONAL
* " EXPORTING
* "     VALUE(NOTIFHEADER_EXPORT) LIKE BAPI2080_NOTHDRE STRUCTURE
* "         BAPI2080_NOTHDRE
* " TABLES
* "     NOTITEM STRUCTURE BAPI2080_NOTITEMI OPTIONAL
* "     NOTIFCAUS STRUCTURE BAPI2080_NOTCAUSI OPTIONAL
* "     NOTIFACTV STRUCTURE BAPI2080_NOTACTVI OPTIONAL
* "     NOTIFTASK STRUCTURE BAPI2080_NOTTASKI OPTIONAL
* "     NOTIFPARTNR STRUCTURE BAPI2080_NOTPARTNRI OPTIONAL
* "     LONGTEXTS STRUCTURE BAPI2080_NOTFULLTXTI OPTIONAL
* "     KEY_RELATIONSHIPS STRUCTURE BAPI2080_NOTKEYE OPTIONAL
* "     RETURN STRUCTURE BAPIRET2 OPTIONAL
* "     NOTIFSTATUS STRUCTURE JSTAT
*-----
* Create service notification
CALL FUNCTION 'BAPI_ALM_NOTIF_CREATE'
EXPORTING
    external_number    = external_number
    noti f_type        = noti f_type
    noti fheader       = noti fheader
    task_determinati on = task_determinati on
    sender             = sender
    order id           = order id
IMPORTING
    noti fheader_export = noti fheader_export
TABLES

```

```

noti tem      = noti tem
noti fcaus    = noti fcaus
noti factv    = noti factv
noti ftask    = noti ftask
noti fpartnr  = noti fpartnr
longtexts    = longtexts
key_relati onshi ps = key_relati onshi ps
return       = return.
CHECK return[] IS INITIAL .

* Save service notification
CALL FUNCTION 'BAPI_ALM_NOTIF_SAVE'
EXPORTING
  number = noti fheader_export-noti f_no.

ENDFUNCTION.

```

Figure 9-7: Customer-Specific Service Function Module: Creation of Service Notification

You create the `Z_BAPI_ALM_NOTIF_TASK_COMPLETE` function module by copying the `BAPI_ALM_NOTIF_TASK_COMPLETE` function module and then delete all the copied ABAP code. With the copy, the custom-specific function module has exactly the same parameter interface as the BAPI. Now all the ABAP code is replaced by a call of the `BAPI_ALM_NOTIF_TASK_COMPLETE` function module (passing all the parameter of the interface) followed by the call of the `BAPI_ALM_NOTIF_SAVE` function module. After the generation of the BAPI migration object, change the **Return Field** on the *Migration Object* maintenance screen to the `AUTO-IMPORT_FIELDS-NUMBER` field. This saves the number of the created service notification in the KSM. After the generation of the BAPI migration object with this function module as the service function module, change the migration Customizing of the `IMPORT_FIELDS-NUMBER` field rule to the *Via KSM* field rule and enter the name of the first migration object as the top object in the **MigObject 1** field. This Customizing ensures, that during data migration the completion step will fail due to a missing entry in the KSM for the first object if the previous process step failed.

```

FUNCTION z_bapi_alm_noti f_task_compl ete.
*-----
*"" Lokale Schnittstelle:
* IMPORTING
*   VALUE(NUMBER) LIKE BAPI 2080_NOTHDRE-NOTIF_NO
*   VALUE(TASK_KEY) LIKE BAPI 2080_NOTTASKE-TASK_KEY
*   VALUE(CARRIED_OUT_BY) LIKE BAPI QMSM-CARRIED_OUT_BY OPTIONAL
*   VALUE(CARRIED_OUT_DATE) LIKE BAPI QMSM-CARRIED_OUT_DATE OPTIONAL
*   VALUE(CARRIED_OUT_TIME) LIKE BAPI QMSM-CARRIED_OUT_TIME OPTIONAL
*   VALUE(LANGU) LIKE BAPI 2080_NOTSTI -LANGU DEFAULT SY-LANGU
*   VALUE(LANGUI SO) LIKE BAPI 2080_NOTSTI -LANGUI SO OPTIONAL
*   VALUE(TESTRUN) LIKE BAPI 20783T-STATUS_IND DEFAULT SPACE
* EXPORTING
*   VALUE(SYSTEMSTATUS) LIKE BAPI 2080_NOTADT-SYSTATUS
*   VALUE(USERSTATUS) LIKE BAPI 2080_NOTADT-USRSTATUS
* TABLES
*   RETURN STRUCTURE BAPI RET2 OPTIONAL
*-----

* Complete task of service notification
CALL FUNCTION 'BAPI_ALM_NOTIF_TASK_COMPLETE'
EXPORTING
  number      = number
  task_key    = task_key
  carried_out_by = carried_out_by
  carried_out_date = carried_out_date
  carried_out_time = carried_out_time
  langu       = sy-langu
  langu so    = langu so
  testrun     = ''
IMPORTING
  systemstatus = systemstatus
  userstatus   = userstatus
TABLES
  return       = return.

```

```

CHECK return[] IS INITIAL .

* Save change of service notification
CALL FUNCTION 'BAPI_ALM_NOTIF_SAVE'
  EXPORTING
    number = number.

ENDFUNCTION.

```

Figure 9-8: Customer-Specific Service Function Module: Completion of Tasks

Create the `Z_BAPI_ALM_NOTIF_CHANGEUSRSTAT` function module by copying the `BAPI_ALM_NOTIF_CHANGEUSRSTAT` function module and then deleting all the copied ABAP code. With the copy, the custom-specific function module has exactly the same parameter interface as the BAPI. Now all the ABAP code is replaced by a call of the BAPI `BAPI_ALM_NOTIF_CHANGEUSRSTAT` function module (passing all the parameter of the interface), followed by the call of the `BAPI_ALM_NOTIF_SAVE` function module. After the generation of the BAPI migration object, change the **Return Field** on the *Migration Object* maintenance screen to `AUTO-IMPORT_FIELDS-NUMBER`. This saves the number of the created service notification in the KSM. After the generation of the BAPI migration object with this function module as the service function module, the migration Customizing of the `IMPORT_FIELDS-NUMBER` field has to be changed to the *Via KSM* field rule and the name of the second migration object is saved as the top object in the **MigObject 1** field. This Customizing ensures, that during data migration this step will fail due to a missing entry in the KSM for the second object if one of the previous process steps failed.

```

FUNCTION z_bapi_alm_notif_changeusrstat.
*-----
* " Interface local :
* " IMPORTING
* "   VALUE(NUMBER) LIKE BAPI 2080_NOTHDRE-NOTIF_NO
* "   VALUE(USR_STATUS) LIKE BAPI 2080_NOTUSRSTATI
* "   STRUCTURE BAPI 2080_NOTUSRSTATI
* "   VALUE(SET_INACTIVE) LIKE BAPI 20783T-STATUS_IND DEFAULT SPACE
* "   VALUE(TESTRUN) LIKE BAPI 20783T-STATUS_IND DEFAULT SPACE
* " EXPORTING
* "   VALUE(SYSTEMSTATUS) LIKE BAPI 2080_NOTADT-SYSTATUS
* "   VALUE(USERSTATUS) LIKE BAPI 2080_NOTADT-USRSTATUS
* " TABLES
* "   RETURN STRUCTURE BAPI RET2 OPTIONAL
*-----

* Chane user status of service notification
CALL FUNCTION 'BAPI_ALM_NOTIF_CHANGEUSRSTAT'
  EXPORTING
    number      = number
    usr_status  = usr_status
    set_inactive = set_inactive
    testrun     = testrun
  IMPORTING
    systemstatus = systemstatus
    userstatus   = userstatus
  TABLES
    return       = return.
CHECK return[] IS INITIAL .

* Save change of service notification
CALL FUNCTION 'BAPI_ALM_NOTIF_SAVE'
  EXPORTING
    number = number.

ENDFUNCTION.

```

Figure 9-9: Customer-Specific Service Function Module: Change of User Status

9.7 Service Function Module to Execute a Program

Chapter 7.12.5 *How to create a migration object to execute an ABAP program?* describes the need for a service function module to submit reports. The code snippet in Figure 9-10 can be used as a basis for an own implementation.

```

FUNCTION Z_MIG_EXECUTE_PROGRAM.
** -----
** "" Local Interface:
** IMPORTING
**     REFERENCE(X_REPORT) TYPE PROGRAMM
**     REFERENCE(X_VARIANT) TYPE VARIANT OPTIONAL
** EXPORTING
**     REFERENCE(RETURN) TYPE BAPIRET2
** TABLES
**     XT_PARAMETER STRUCTURE RSPARAMS OPTIONAL
** -----
*
* X_REPORT
*   Name of the report to be executed
* X_VARIANT
*   Name of a variant that must exists for the report
* XT_PARAMETER
*   SELNAME must contain the name of a parameter or selection criterion
*   for the selection screen in block capitals.
*   KIND must contain the type of selection screen component (P for
*   parameters, S for selection criteria).
*   SIGN, OPTION, LOW, and HIGH must contain the values specified
*   for the selection table columns that have the same names as the
*   selection criteria; in the case of parameters, the value must be
*   specified in LOW and all other components are ignored.
*
* Submit Report
IF NOT x_variant IS INITIAL.
    SUBMIT (x_report) AND RETURN USING SELECTION-SET x_variant.
ELSE.
    SUBMIT (x_report) AND RETURN WITH SELECTION-TABLE xt_parameter.
ENDIF.

* check error
IF sy-subrc <> 0.
    CALL FUNCTION 'BALW_BAPIRETURN_GET2'
        EXPORTING
            type   = sy-msgty
            cl     = sy-msgid
            number = sy-msgno
            par1   = sy-msgv1
            par2   = sy-msgv2
            par3   = sy-msgv3
            par4   = sy-msgv4
        IMPORTING
            return = return.
ENDIF.
ENDFUNCTION.

```

Figure 9-10: Function Module to Execute Programs in Parallel

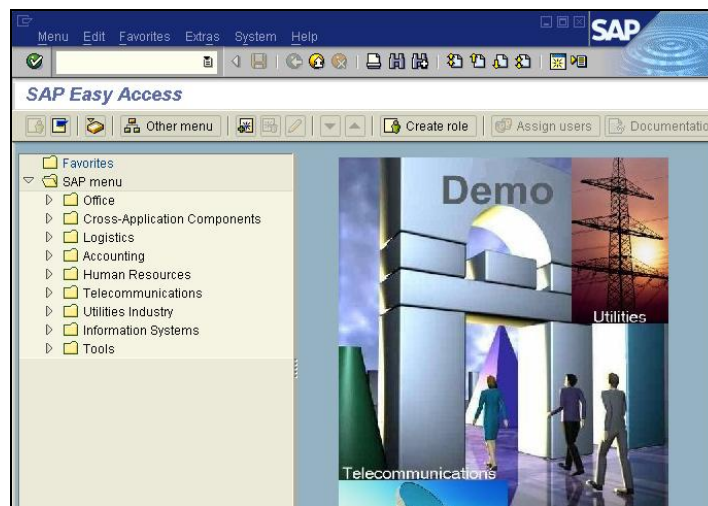
10 Getting Started

10.1 Motivation

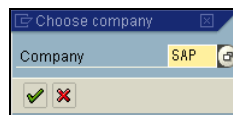
This chapter explains step-by-step how to get started with the IS Migration Workbench. It shows how to create a migration company, setup the user parameters and how to maintain the Customizing of migration objects. Subsequently a sample import file is created followed by the execution of a data import. The result of the data import is checked, the errors in the import file corrected and the import file re-imported to successfully import all data objects in the created import file.

10.2 Start of the IS Migration Workbench

1. Enter the transaction code EMIGALL in the command field



2. Confirm the popup in pushing the *Continue* button.



3. The main screen of the IS Migration Workbench is shown.

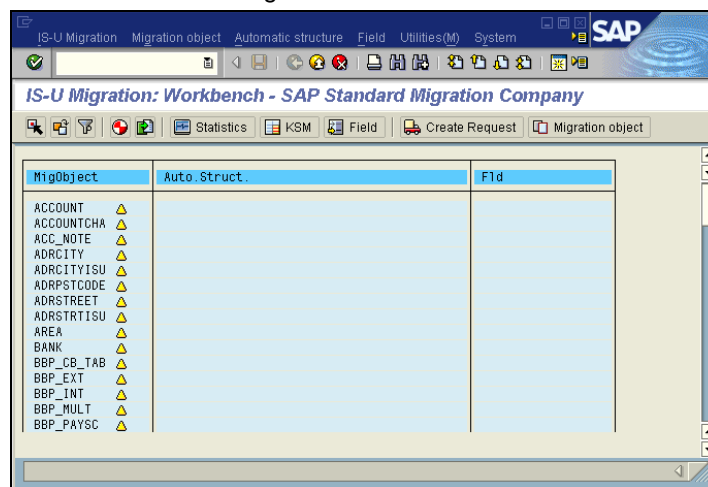


Figure 10-1: Procedure to Start the IS Migration Workbench

10.3 Creation of a Migration Company

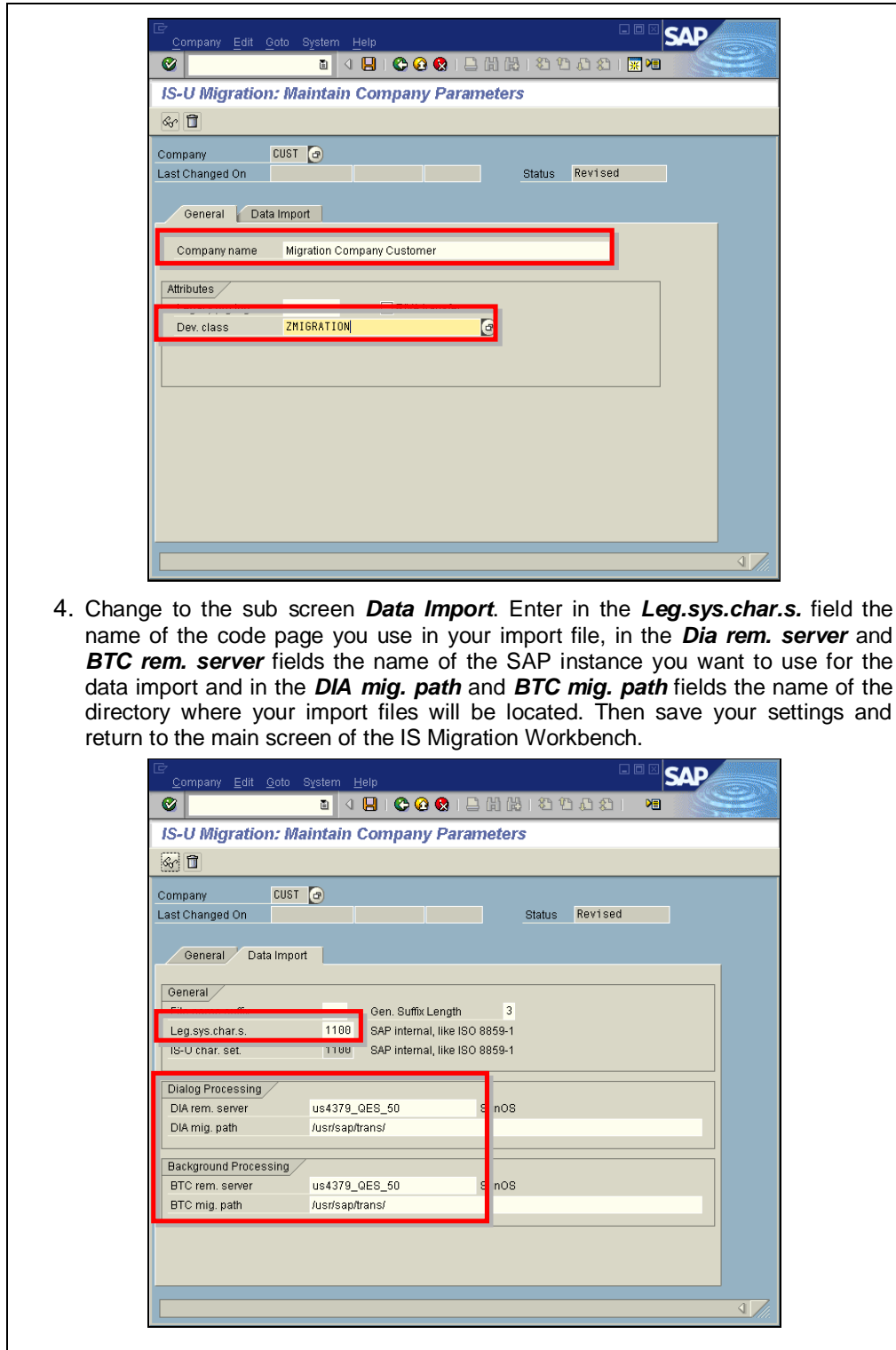
1. Choose *IS-U Migration* → *Company Maintenance*

The screenshot shows the SAP IS-U Migration: Display Company Parameters screen. The 'Company' field is set to 'SAP'. The 'Company name' field contains 'SAP Standard Migration Company'. The 'Dev. class' field contains 'EEMI'. The 'Attributes' section shows 'Legacy prg. lng.' and 'RIVA transfer' (unchecked). The 'Data Import' tab is selected.

2. Enter the name of your migration company in the **Company** field and push the **Create** button

The screenshot shows the SAP IS-U Migration: Display Company Parameters screen. The 'Company' field is set to 'CUSTJ'. The 'Company name' field contains 'SAP Standard Migration Company'. The 'Dev. class' field contains 'EEMI'. The 'Attributes' section shows 'Legacy prg. lng.' and 'RIVA transfer' (unchecked). The 'Data Import' tab is selected. A red box highlights the 'Create' button in the top left corner.

3. Enter in the **Company name** field a description of your migration company and a project specific migration class in the **Dev.class** field.



4. Change to the sub screen **Data Import**. Enter in the **Leg.sys.char.s.** field the name of the code page you use in your import file, in the **Dia rem. server** and **BTC rem. server** fields the name of the SAP instance you want to use for the data import and in the **DIA mig. path** and **BTC mig. path** fields the name of the directory where your import files will be located. Then save your settings and return to the main screen of the IS Migration Workbench.

Figure 10-2: Procedure to Create a Migration Company

10.4 Creation of a Migration User

1. Choose *IS-U Migration* → *User Parameters*.

The screenshot shows the 'IS-U Migration: Display User Parameters' dialog box. At the top, there is a menu bar with 'User', 'Edit', 'Goto', 'System', and 'Help'. Below the menu bar is a toolbar with various icons. The main area of the dialog box is divided into several sections. The 'User' section at the top shows 'User' as 'KELLERFR', 'Last changed by' as 'KELLERFR', '27.04.2009 13:47:37', and 'Status' as an empty field. Below this are four tabs: 'General', 'Data import', 'Data exchange', and 'Monitoring'. The 'General' tab is selected. It contains two sub-sections: 'Company' and 'Display'. The 'Company' section has a 'Company' field with 'SAP' selected, a checkbox for 'Fxd start co.' which is unchecked, and a checkbox for 'Other disconn. sts permttd' which is unchecked. The 'Display' section has four checkboxes: 'No Longer Display' (unchecked), 'Display WB status' (unchecked), 'Display entire field' (unchecked), and 'Barrier-Free Display' (unchecked). Below these checkboxes is a 'Number displayed DRs' field with the value '60'.

2. Push the *Change* button.

The screenshot shows the 'IS-U Migration: Maintain User Parameters' dialog box. It has the same menu bar and toolbar as the previous dialog box. The main area is divided into several sections. The 'User' section at the top shows 'User' as 'KELLERFR', 'Last changed by' as 'KELLERFR', '27.04.2009 13:47:37', and 'Status' as 'Revised'. Below this are four tabs: 'General', 'Data import', 'Data exchange', and 'Monitoring'. The 'General' tab is selected. It contains two sub-sections: 'Company' and 'Display'. The 'Company' section has a 'Company' field with 'SAP' selected, a checkbox for 'Fxd start co.' which is unchecked, and a checkbox for 'Other disconn. sts permttd' which is unchecked. The 'Display' section has four checkboxes: 'No Longer Display' (unchecked), 'Display WB status' (unchecked), 'Display entire field' (unchecked), and 'Barrier-Free Display' (unchecked). Below these checkboxes is a 'Number displayed DRs' field with the value '60'.

3. Enter in the **Company** field the name of your migration company and mark the **Display WB status** and **Display entire field** fields.

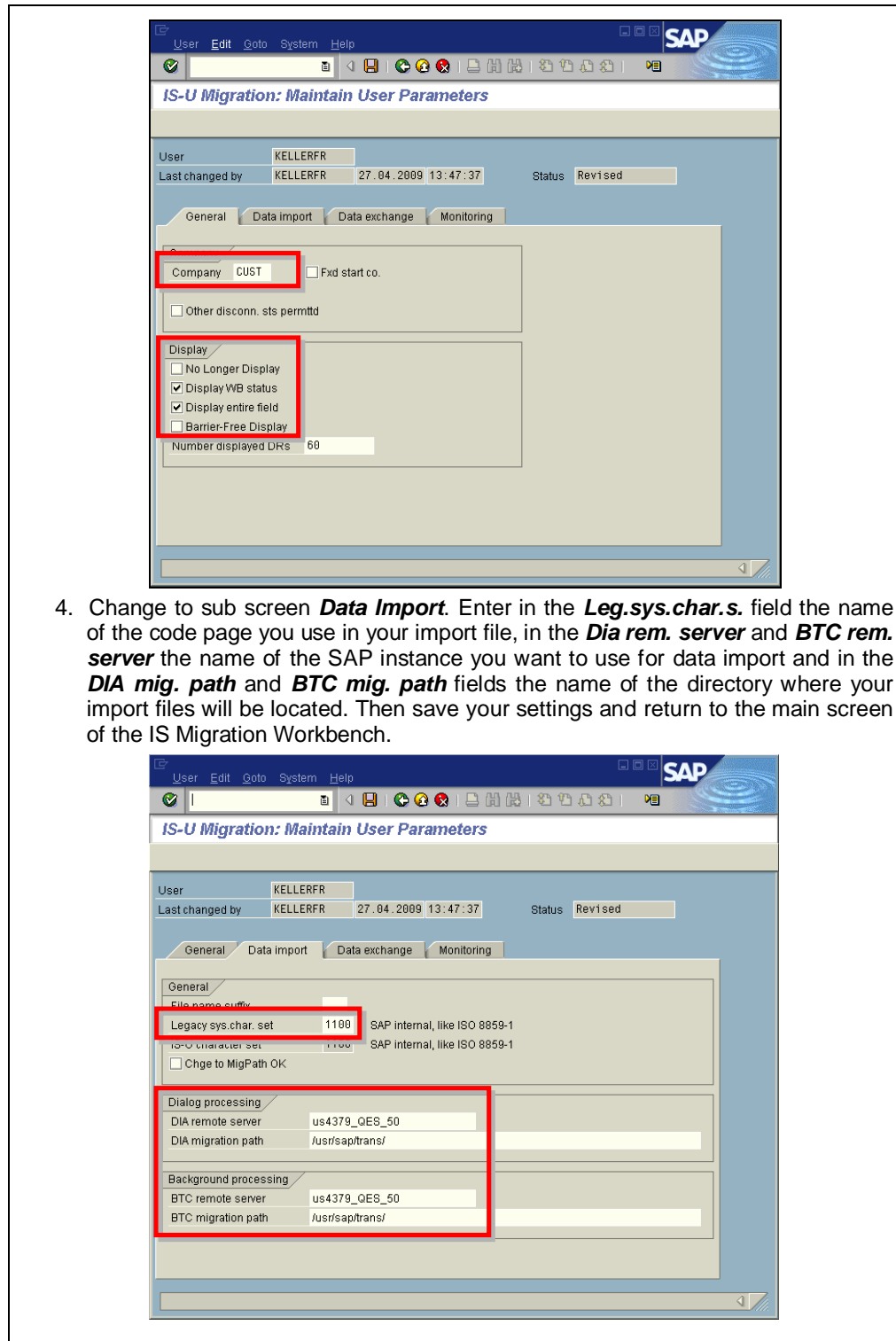
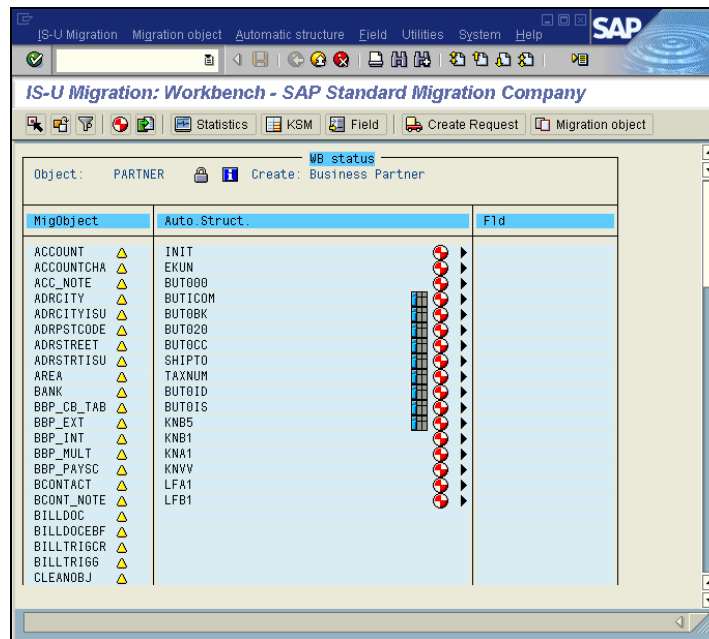


Figure 10-3: Procedure to Create a Migration User

10.5 Copy of a Migration Object

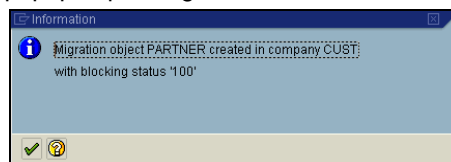
1. Double-click on the PARTNER migration object to mark the object. Then push the *Copy Migration Object* button



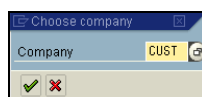
2. Enter in the **Company** field the name of your migration company. Then push the *Copy* button.



3. Confirm the popup in pushing the *Continue* button.



4. Choose *IS-U Migration → Other Company* or push the *Other Company* button.



5. Enter in the **Company** field the name of your migration company. Then push the *Continue* button to display the copied PARTNER migration object in your migration company.

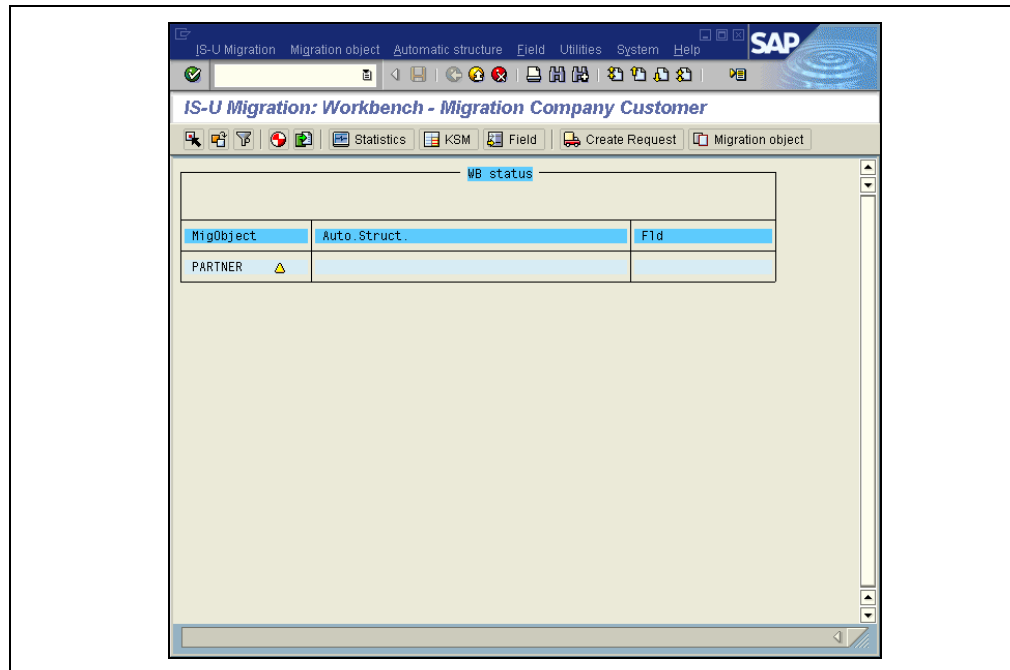
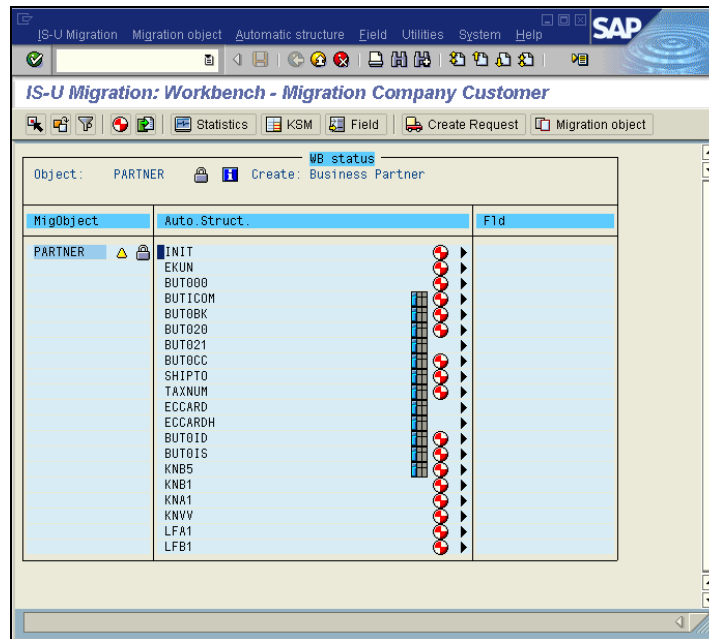


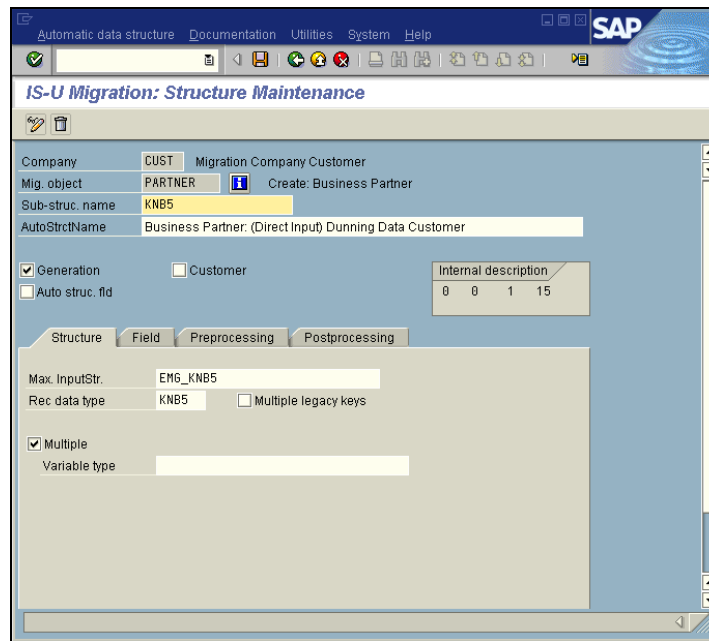
Figure 10-4: Procedure to Copy a Migration Object

10.6 Maintenance of the Structure Customizing

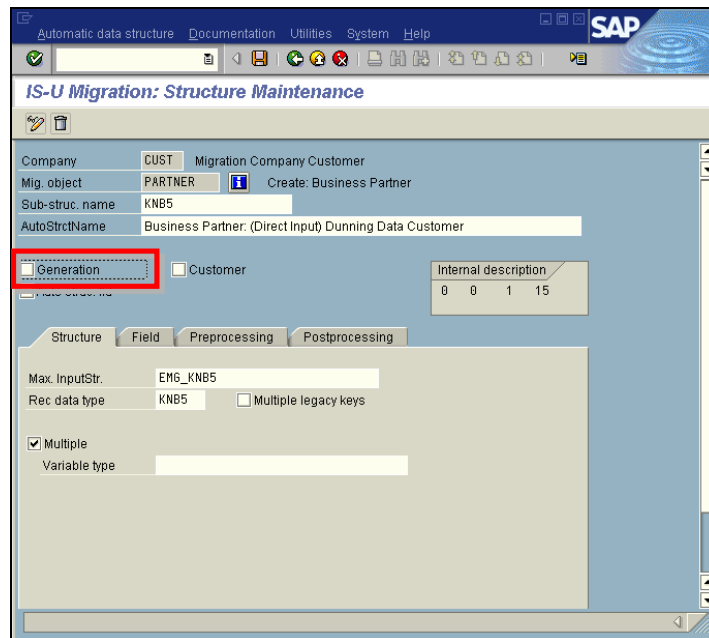
1. Double-click on the PARTNER migration object to block this migration. Then choose *Automatic structure* → *Display list* → *All*. The KNB5 structure is marked with the icon. This icon indicates that the KNB5 structure is marked to be used in the migration process.



2. Double-click on the KNB5 structure. Then chose Automatic Structure → Change.



3. Unmark the **Generation** field because the KNB5 structure will not be required in our migration process.



4. Save the changed Customizing and return to the main screen of the IS Migration Workbench. The KNB5 structure is not longer marked with the icon.

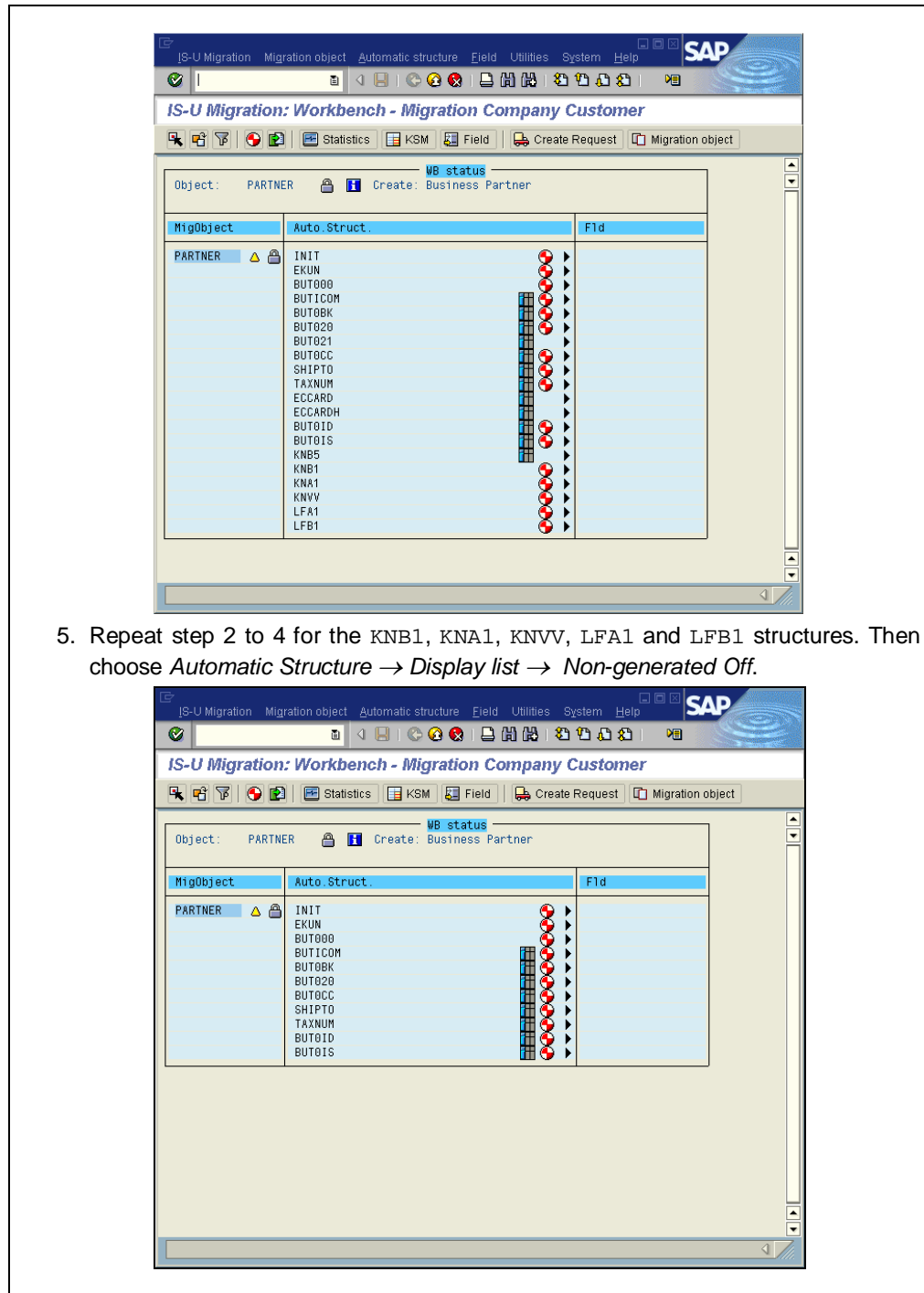
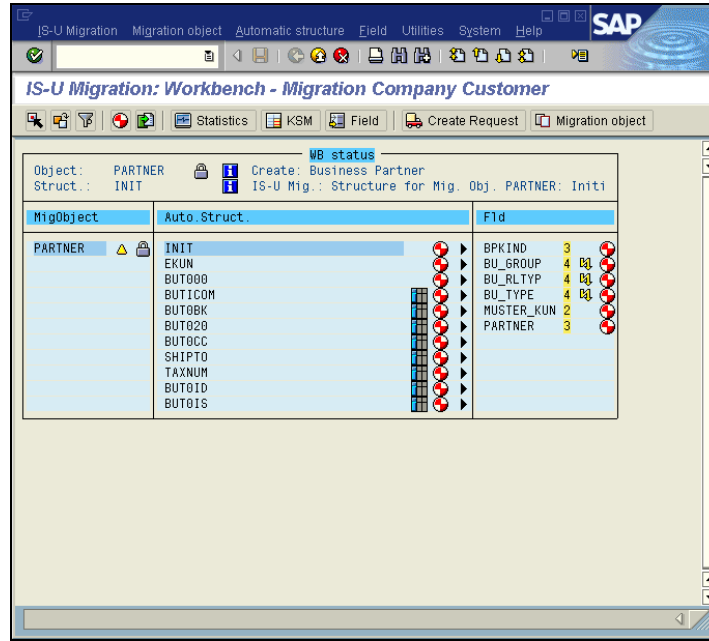


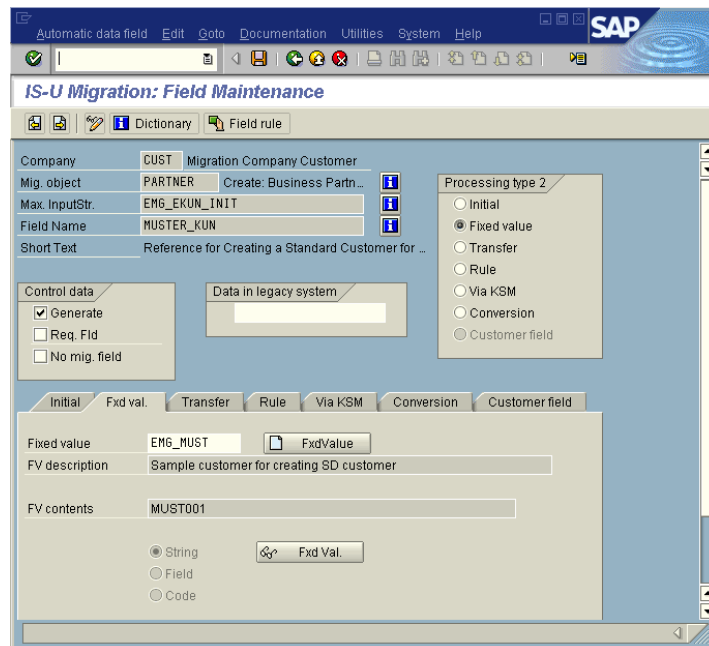
Figure 10-5: Procedure to Maintain the Structure Customizing of a Migration Object

10.7 Maintenance of the Field Customizing

1. Double-click on the PARTNER migration object to block this object. Then double-click on the INIT structure. The MUSTER_KUN field is marked with the icon. This icon indicates that the MUSTER_KUN field is presently marked as to be used in the migration process.



2. Double-click on the MUSTER_KUN field.

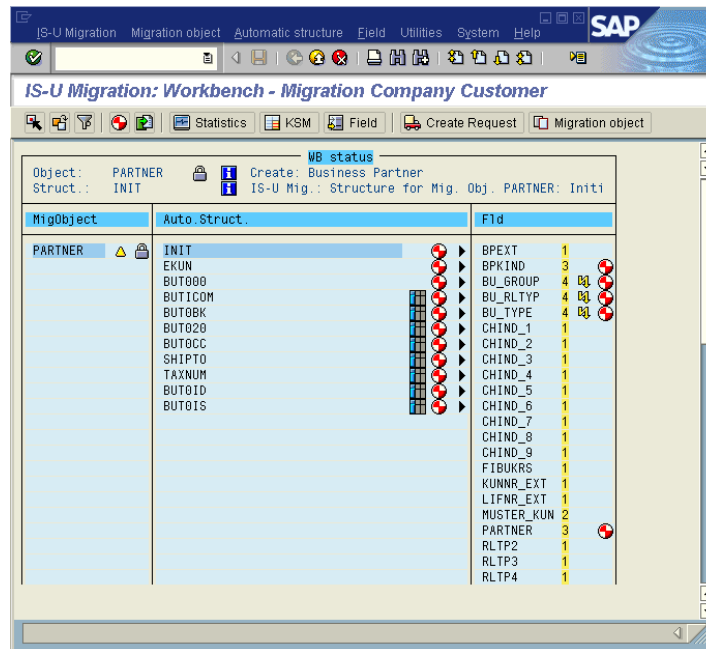


3. For example, unmark the **Generate** field because the MUSTER_KUN field will not be required in our migration process. You can also change the field rule of the field or other field relevant Customizing.

4. Save the changed Customizing and return to the main screen of the IS Migration Workbench. The MUSTER_KUN field is not displayed because only the fields are displayed which are marked as to be generated.

MigObject	Auto. Struct.	Fld
PARTNER	INIT	BPKIND 3
	EKUN	BU_GROUP 4
	BUT000	BU_RLTYP 4
	BUTICOM	BU_TYPE 4
	BUT0BK	PARTNER 3
	BUT020	
	BUT0CC	
	SHIPTO	
	TAXNUM	
	BUT010	
	BUT01S	

5. Choose *Field* → *Display list* → *Non-generated On* to display all fields of the selected structure.

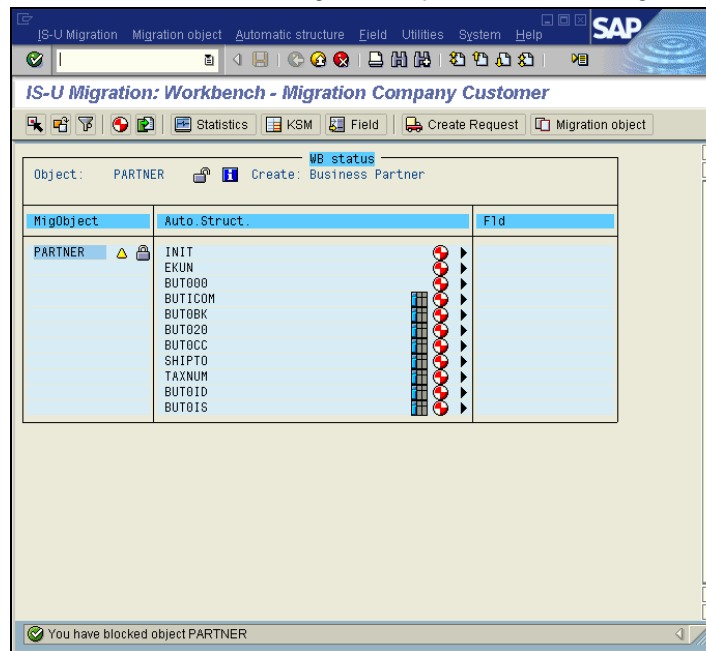


6. Repeat step 1 to 4 for all fields you want to change the Customizing.

Figure 10-6: Procedure to Maintain the Field Customizing of a Migration Object

10.8 Creation of an Import File

1. Double-click on the PARTNER migration object to block this migration object.



2. Choose *Migration Object* → *Change* to verify, if the blocking status of the migration object allows a data import. If needed change the blocking status to a status that allows a data import (for example, 000).

IS-U Migration: Object Maintenance

Company: CUST Migration Company Customer

Mig. object: PARTNER Mig. class: PARTNER

Mig. obj. text: Create: Business Partner

Mig. obj. shbr: PAR 3 Create

Blocking Status: 000 Do not block any

Object Data Relationship data Parameters

Service module: ISU_M_PARTNER_CREATE_DARK Interface

Auto struc. type: ISUMI_PARTNER_AUTO Mult. leg. keys

Return structure: NEW_PAR ☒ Structure type BUT000

☐ Internal Table ☐ Structure type

Return field: NEW_PAR-PARTNER

3. Choose *Migration Object* → *Generate Report* or push the **Generate Report** button if the load report of the migration object needs to be generated as indicated by an icon. The icon must change to icon after a load report has been generated successfully.

IS-U Migration Migration object Automatic structure Field Utilities System Help

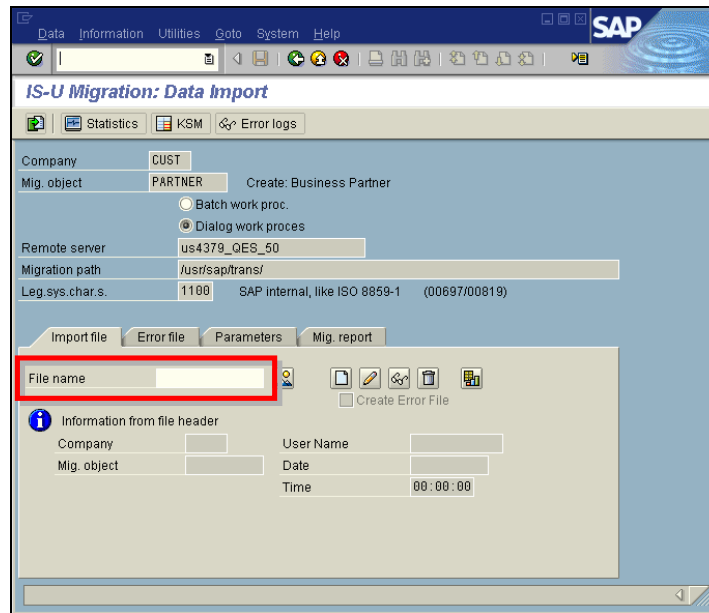
IS-U Migration: Workbench - Migration Company Customer

Object: PARTNER Create: Business Partner

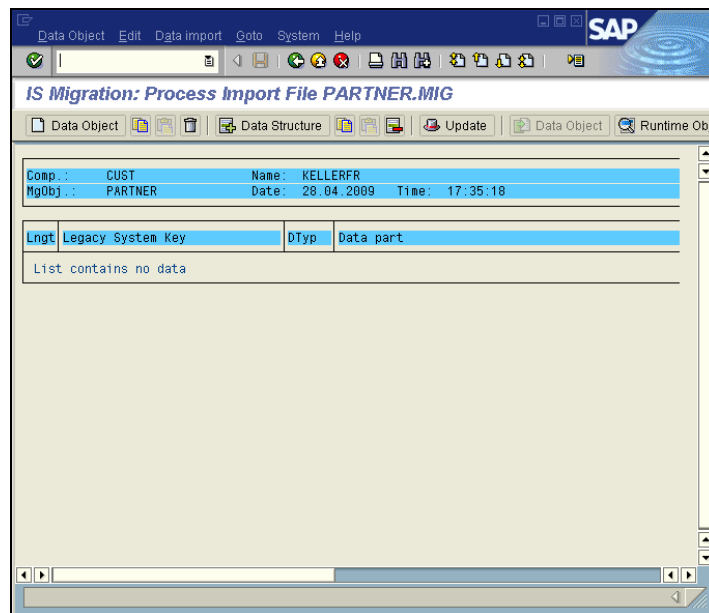
MigObject	Auto. Struct.	Fld
PARTNER	INIT	
	EKUN	
	BUT000	
	BUTICOM	
	BUT0BK	
	BUT020	
	BUT0CC	
	SHIPT0	
	TAXNUM	
	BUT01D	
	BUT01S	

RE3TR_CUST_PAR was generated

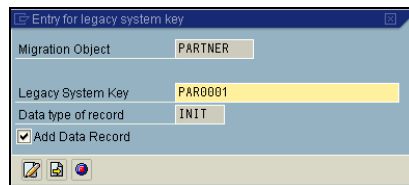
4. Choose *Migration Object* → *Data Import* or push the **Data Import** button.



5. Enter in the **File name** field the name of the of the import file to be created. Then push the **Create Import File** button and the import file editor is called.



6. Push the **Create Data Object** button to create a new data object in the import file and enter in the Legacy System Key field the legacy system of the new data object. Then push the **Specify Data** button.



7. Enter the data in the fields of the INIT structure. Push the **F4** key if you need a search help to choose an appropriate value for a field. Push the **Continue** button after you filled all required fields.

IS-U Migration: Display Data Record in File

Mig. object: PARTNER
 LegSystem key: PAR0001
 Rec. data type: INIT IS-U Mig.: Structure for Mig. Obj. PARTNER: Initial Data

Field Na	T	A...	OutLe...	Short Description	Data	No. of ...
PARTNER	3	C	10	Business Partner Number		10 0
BU_RLTYP	4	C	6	BDT: Object part	MKK	6 0
BU_TYPE	4	C	1	Business partner category	2	1 0
BU_GROUP	4	C	4	Business Partner Grouping	0001	4 0
BPKIND	3	C	4	Business Partner Type	0001	0 0

Migration object Automatic structure Field

8. Push either the *Continue* button to enter data for the EKUN structure, or the *Next Structure* button to change to the next structure. If you have entered all relevant data for the data object push the *Complete Entry* button to finish the creation of the data object.

Entry for legacy system key

Migration Object: PARTNER
 Legacy System Key: PAR0001
 Data type of record: EKUN
☒ Add Data Record

9. After you have pushed the Complete Entry button, the import file editor shows the newly created data object.

SAP

IS Migration: Process Import File PARTNER.MIG

Comp.: CUST Name: KELLERFR
 MgObj.: PARTNER Date: 28.04.2009 Time: 17:51:49

Lngr	Legacy System Key	DTyp	Data part
003F	PAR0001	INIT	MKK 2 00001
0056	PAR0001	EKUN	
019C	PAR0001	BUT000	MOVIE CORPORATION Movie Corp 0002
004E	PAR0001	BUT00K	0001US 123123123 123872456812
00C9	PAR0001	BUT020	New York City 10004
00C9	PAR0001	BUT020	
00C9	PAR0001	&ENDE	
FFFF			

10. Repeat step 6 – 9 to create more data objects in the import file.

SAP

IS Migration: Process Import File PARTNER.MIG

Comp.: CUST Name: KELLERFR
MigObj.: PARTNER Date: 28.04.2009 Time: 17:55:36

Lngr	Legacy System Key	DTyp	Data part
003F	PAR0001	INIT	MKK 2 00001
0056	PAR0001	EKUN	
019C	PAR0001	BUT000	MOVIE CORPORATION Movie Corp 0002
004E	PAR0001	BUT000	0001US 123123123 123872456812 Cmp The
00C9	PAR0001	BUT020	New York City 10004
00C9	PAR0001	BUT020	
00C9	PAR0001	&ENDE	
003F	PAR0002	INIT	MKK 00001
0056	PAR0002	EKUN	
019C	PAR0002	BUT000	LOTHAR Matthaeus Mr. 0001
00C9	PAR0002	BUT020	New York City 10004
00C9	PAR0002	BUT020	
00C9	PAR0002	&ENDE	
003F	PAR0003	INIT	MKK 00001
0056	PAR0003	EKUN	
019C	PAR0003	BUT000	WILL Smith MR 0001
004E	PAR0003	BUT000	0001US 123123123 123872456812
00C9	PAR0003	BUT020	New York City 10004
00C9	PAR0003	BUT020	
00C9	PAR0003	&ENDE	
003F	PAR0004	INIT	MKK 00001
0056	PAR0004	EKUN	
019C	PAR0004	BUT000	JONES Jones MR 0001
00C9	PAR0004	BUT020	New York City 10004
00C9	PAR0004	BUT020	
00C9	PAR0004	&ENDE	
FFFF			

11. Push the *Back* button to leave the import file editor once you have added all data objects to the import file.

SAP

IS-U Migration: Data Import

Company: CUST
Mig. object: PARTNER Create: Business Partner

☐ Batch work proc.
☒ Dialog work proces

Remote server: us4379_QES_50
Migration path: /usr/sap/trans/

Leg.sys.char.s: 1100 SAP internal, like ISO 8859-1 (00697/00819)

Import file | Error file | Parameters | Mig. report

File name: PARTNER.MIG

☐ Create Error File

Information from file header

Company: CUST User Name: KELLERFR
Mig. object: PARTNER Date: 28.04.2009
Time: 17:55:36

Figure 10-7: Procedure to Create an Import File

10.9 Execution of a Data Import

1. Double-click on the PARTNER migration object to block this object. Then choose *Migration Object* → *Data Import* or push the **Data Import** button.

The screenshot shows the 'IS-U Migration: Data Import' dialog box. The 'File name' field is highlighted with a red rectangle. The dialog box contains the following fields and options:

- Company: CUST
- Mig. object: PARTNER
- Create: Business Partner
- Batch work proc. (radio button)
- Dialog work proces (radio button)
- Remote server: us4379_QES_50
- Migration path: /usr/sap/trans/
- Leg.sys.char.s: 1100
- SAP internal, like ISO 8859-1 (00697/00819)
- Import file (tab)
- Error file (tab)
- Parameters (tab)
- Mig. report (tab)
- File name: (empty field)
- Create Error File (checkbox)
- Information from file header (info icon)
- Company: (empty field)
- User Name: (empty field)
- Mig. object: (empty field)
- Date: (empty field)
- Time: 00:00:00

2. Enter in the **File name** field the name of the import file and push the *Return* key. Choose either **Batch work proc.** for a data import in background or **Dialog work process** for an online data import.

The screenshot shows the 'IS-U Migration: Data Import' dialog box with the 'File name' field filled with 'PARTNER.MIG'. The dialog box contains the following fields and options:

- Company: CUST
- Mig. object: PARTNER
- Create: Business Partner
- Batch work proc. (radio button)
- Dialog work proces (radio button)
- Remote server: us4379_QES_50
- Migration path: /usr/sap/trans/
- Leg.sys.char.s: 1100
- SAP internal, like ISO 8859-1 (00697/00819)
- Import file (tab)
- Error file (tab)
- Parameters (tab)
- Mig. report (tab)
- File name: PARTNER.MIG
- Create Error File (checkbox)
- Information from file header (info icon)
- Company: CUST
- User Name: KELLERFR
- Mig. object: PARTNER
- Date: 28.04.2009
- Time: 17:55:36

3. Push either the *Import Data* button or *Data* → *Import data* to start the data import.
4. Push the *Statistics* button to check the result of the data import.

User Name	Se De Comp	Mig.object	SPDate	Key Time	SID Proc	Ru St	Full	Error	Throughput	rec/h
KELLERFR	CUST	PARTNER	28.04.2009	18:58:26	001	DIALOG	✓	4	1	1.800

- The statistics shows that four data objects have been processed but one of them with an error. Push the button to display the error log. The error message indicates a problem with the transferred value 0000 in the BU_GROUP field (*Business Partner Grouping*) in the data object with the legacy system key. The BU_GROUP field is part of the INIT structure.

Company	Migration Object
CUST	PARTNER

Message Text	LTxt
Grouping 0000 does not exist	
Error during processing of legacy system key PAR0002	

- Return to the import file editor. Display the INIT structure of the data object with the PAR0001 legacy system key and put the cursor in the BU_GROUP field. Then push the F4 key to display the search help.

Grp.	Short name
0001	
0002	Ext.No.Assignmnt
0003	
0004	
0005	
DAR1	Loans
GPEX	Ext.no.assignmnt
GPIN	Int.no.assignmnt
IMMO	Real Estate
MDM0	
SRM	Stakeholder
TR01	TreasuryPartner
TR02	TreasuryPartner

13 Entries found

7. There is no value 0000 defined for business partner grouping. Correct the value in the import file from 0000 to 0001.

Field Na	T	A...	OutLe...	Short Description	Data	No. of...
PARTNER	3	C	10	Business Partner Number		10
BU_RLTYP	4	C	6	BDT: Object part	MKK	6
BU_GROUP	4	C	4	Business Partner Grouping	0001	1

8. Push the *Back* button to leave the import file editor. Then change to the **Parameters** sub screen and mark the **Restart** field to suppress the error message EM 101 *Legacy system key & 1 has already been migrated* in the error log for the three data objects which have already been migrated.

IS-U Migration: Data Import

Company: CUST
 Mig. object: PARTNER
 Create: Business Partner
 Batch work proc.
 Dialog work proces
 Remote server: us4379_QES_50
 Migration path: /usr/sap/trans/
 Leg. sys. char.s.: 1100 SAP internal, like ISO 8859-1 (00697/00819)

Import file | Error file | **Parameters** | Mig. report

Commit control
 Commit interval: 1

Cancellation control
 Absolute error:
 Rel. error in %:
 Error interval:

Other parameters
☒ Restart
☐ Test run
☐ Check file layout
☒ Online Update
☒ W/o screens

9. Push either the *Import Data* button or *Data* → *Import data* to start the data import.
10. Push the *Statistics* button to check the result of the data import.

IS-U Migration: Analysis of Migration Statistics

Display | [Icons] | Total Throughput | Imported | Reset | Since Start | Statistics | File Information

User Name																
Se	De	Comp.	Mig.	object	SPDate	Key	Time	SID	Proc.	Ru	St	Full	Error	Throughput	rec/h	
KELLERFR																
				CUST	PARTNER	28.04.2009	19:20:58	001	DIALOG				4	0		514
				CUST	PARTNER	28.04.2009	18:58:26	001	DIALOG				4	1		1.800
														total		

11. The statistics shows that all four data objects have been migrated successfully.
12. Push the Back button to leave the statistic screen.

The screenshot shows the SAP IS-U Migration: Data Import dialog box. The title bar includes menu items: Data, Information, Utilities, Goto, System, Help. The main window has tabs for Statistics, KSM, and Error logs. The 'Import file' tab is active, showing fields for File name (PARTNER.MIG), User Name (KELLERFR), Date (28.04.2009), and Time (19:06:42). There are also options for Batch work proc., Dialog work proces, Remote server (us4379_QES_50), Migration path (/usr/sap/trans/), and Leg.sys.char.s. (1100). A checkbox for 'Create Error File' is present.

File name	User Name	Date	Time
PARTNER.MIG	KELLERFR	28.04.2009	19:06:42

13. Push the *KSM* button to change to the KSM display

Key Allocation System Help

IS Migration: Key and Status Management

Key Allocation Key Allocation Key Allocation

Company: CUST

Migration Object: PARTNER

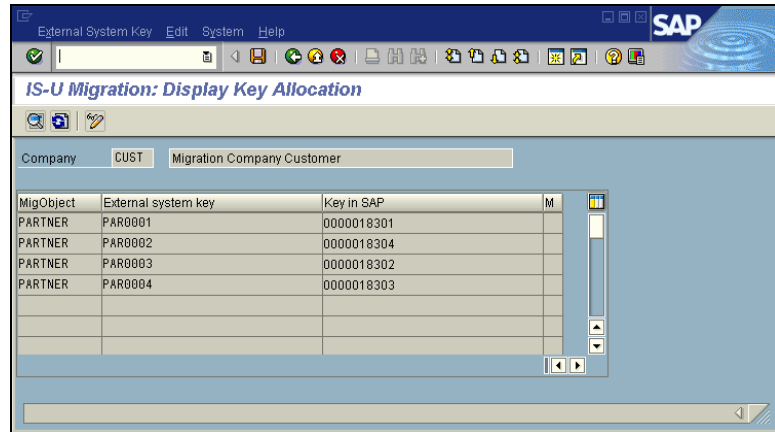
Selection

☒ External system key To

☐ System key IS-UCCS To

Display no. of data records 500

14. Push the *Key allocation* button to display the list of the KSM entries.



15. Position the cursor on one line and push the *Display runtime object* button to display the migrated business partner. Alternatively, double-click on the selected line.

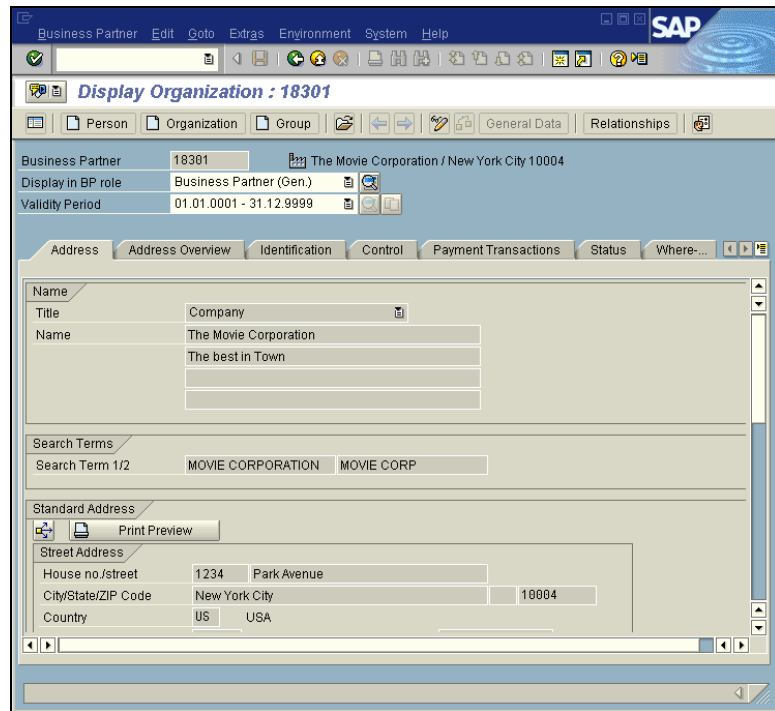


Figure 10-8: Procedure to Import Data