



Peregrine | AssetCenter
Migration



© Copyright 2002 Peregrine Systems, Inc.

All Rights Reserved.

Information contained in this document is proprietary to Peregrine Systems, Incorporated, and may be used or disclosed only with written permission from Peregrine Systems, Inc. This manual, or any part thereof, may not be reproduced without the prior written permission of Peregrine Systems, Inc. This document refers to numerous products by their trade names. In most, if not all, cases these designations are claimed as Trademarks or Registered Trademarks by their respective companies.

Peregrine Systems® and AssetCenter® are trademarks of Peregrine Systems, Inc. or its subsidiaries.

This document and the related software described in this manual are supplied under license or nondisclosure agreement and may be used or copied only in accordance with the terms of the agreement. The information in this document is subject to change without notice and does not represent a commitment on the part of Peregrine Systems, Inc. Contact Peregrine Systems, Inc., Customer Support to verify the date of the latest version of this document.

The names of companies and individuals used in the sample database and in examples in the manuals are fictitious and are intended to illustrate the use of the software. Any resemblance to actual companies or individuals, whether past or present, is purely coincidental.

This product contains software components developed by Apache Software Foundation (<http://www.apache.org>).


This edition applies to version 4.1.0 of the licensed program

AssetCenter

Peregrine Systems, Inc.
Worldwide Corporate Campus and Executive Briefing Center
3611 Valley Centre Drive San Diego, CA 92130
Tel 800.638.5231 or 858.481.5000
Fax 858.481.1751
www.peregrine.com



Table of Contents

Introduction (Migration)	11
Why migrate?	11
What does migration involve?	11
Who is migration intended for?	12
How to use this guide	13
Chapter 1. Supported environments	15
Chapter 2. Overview	17
Chapter 3. Step-by-step migration - preparation phase	25
Preliminary analysis	25
Launching the migration project	26
Training the users and support technicians	27
Preparing your conversion computer	28
Preparing the DBMS server	30
Chapter 4. Step-by-step migration - simulating the conversion of the working database	31
 Verify the integrity of the working database	32

2	Manually adjust the working database	32
3	Propagate database-structure changes	42
4	Copy (1) the working database	46
5	Convert the backup (1) of the working database	48
	Adapting the migration.xml conversion file	48
	Converting the backup (1) of the database	49
	Information about the conversion	50
6	Verify the integrity of the backup (1) of the working database	59
7	Validate the backup (1) of the converted working database.	59
8	Restriction of certain rights on the working database	60
9	Export application data to be manually converted	60
	Processing application data to be manually converted	62
	7 Verify and correct the application data to be manually converted	63
	11 Restore corrected application data	72
	12 Verify the integrity of the backup (1) of the working database	74
	13 Verify restored application data	75
14	Adapt the integration with external tools	76
Chapter 5. Step-by-step migration - converting the working database		77
.		
15	Verify the integrity of the working database	77
16	Block and save the backup (2) of the working database	78
17	Convert the backup (2) of the blocked, working database	78
18	Restore the manually converted application data	79
19	Verify the integrity of the backup (2) of the working database	79
20	Finalize the backup (2) of the working database	80
	Finalization concerning all versions of the source database	80
	Finalization concerning only those source database versions later than version 4.0.0	92
Chapter 6. Step-by-step migration - final phase		99
	Updating AssetCenter programs	99
	Install AssetCenter Server on an administration machine	100
	Delete the AssetCenter caches of your previous database	100
	Upgrade AssetCenter programs	100
	Verify that AssetCenter can be launched without problems	104
	Remove the old connections to databases and create new ones	104
	Modify the customizations of AssetCenter at the level of the client machines if you consider this to be useful	104
	Putting the copy (2) of the converted database into production	104

Chapter 7. Glossary (Migration)	107
Migration	107
Updating AssetCenter programs	107
Converting the working database	108
Conversion file	108
Conversion machine	108
Working database	109
Trigger	109
Chapter 8. References (Migration)	111
Adapting the migration.xml conversion file	111
Warning	111
Reminders	112
What does the conversion file do?	112
Conversion rules	113
Syntax of the conversion file	113
Using special characters	122
Dividing the fields of an previous table between several new tables	123
Transferring a character to a field	124
Converting a field that stores application to be manually converted	126
Using joins	126
Populating foreign keys	127
Dividing source tables between two or more destination tables	127
Converting a numeric string into a text string	128
Manually converting application data	128
SQL commands generated from a conversion file	129
Verifying the conversion file before using it	129
Structural modification made to the database between versions	130
Application data to be manually converted	133
Application data stored in the AssetCenter Script Analyzer database	134
Other application data to verify	137
Structural parameters of the database	137
Other documentation (Migration)	138

List of Figures

2.1. Conversion - process	19
4.1. Propagating structural changes - process	43
4.2. AssetCenter Script Analyzer - *.xml file analysis window	67
4.3. AssetCenter Script Analyzer - script analysis window	70

List of Tables

4.1. Fields that must not contain the ^ character - list	35
4.2. AssetCenter Script Analyzer - menus	65
8.1. Application data to be manually converted - list	134
8.2. Structural parameters of the database - list	137
8.3. Other documentation (Migration) - list	138

Introduction (Migration)

Why migrate?

Version 4 of AssetCenter has changed quite considerably with new structural modifications:

- The database structure (tables, fields, links, indexes) has been modified.

The organization of **Categories/Products/Assets** has been replaced by an organization of **Natures/Models/Assets, Batches Portfolio items and Catalog products**.

- New functions have been added.

All these changes have made it necessary to methodically migrate your earlier version of AssetCenter to the 4.1.0 version.

What does migration involve?

Migration involves performing the following tasks:

- Converting the AssetCenter database to the new format (structure and content).
- Upgrading the AssetCenter programs to the version 4.1.0.

Who is migration intended for?

Migration is intended for any company using a version of AssetCenter earlier than the 4.1.0 version and who want to use the 4.1.0 version.

This migration is performed by the engineers in charge of:

- Administering the AssetCenter database.
 - Installing AssetCenter.
 - Deploying AssetCenter.
-

Warning:

Migration is a complex process that requires:

- A thorough understanding of the earlier version of AssetCenter and of the version 4.1.0 (installation, configuration of parameters, database structure, functions, etc.).
 - Preparation
 - Technical competences
 - Methodology
 - Time
 - Resources
-

How to use this guide

 **Tip:**

Before reading this guide, we recommend that you read some of the other AssetCenter 4.1.0 guides:

- **Installation**
 - **Release Notes**
 - **readme.txt**
 - **Differences between the versions 3.x and 4.1.0**
-
-

 **Tip:**

We also recommend that you read this guide in its entirety and in its presented order.

Chapter Supported environments.

This chapter contains the list of environments supported by the migration.

Read this chapter to make sure your configuration is supported.

Chapter Overview.

This chapter provides an overview of the migration.

Read this chapter to learn about how migration works in general.

Chapter Step-by-step migration - preparation phase.

Chapter Step-by-step migration - simulating the conversion of the working database.

Chapter Step-by-step migration - converting the working database.

Chapter Step-by-step migration - final phase.

These chapters outline the steps involved in the migration.

Start by reading these chapters in their entirety to familiarize yourself with all the steps you will need to perform throughout the migration process.

Then continue, step by step, in the order presented in this guide and paying attention to each detail.

Chapter Glossary (Migration).

This chapter defines the key terms used in migration.

Read this chapter to learn the terminology used in this guide.

Chapter References (Migration).

This chapter contains exhaustive and systematic reference information.

Read this chapter to obtain advanced or supplementary information.

1 Supported environments

CHAPTER

Operating systems and DBMSs

This migration works with all operating systems and DBMSs supported by AssetCenter, except the SQL Anywhere runtime.

To learn which operating systems and DBMSs are supported, refer to the compatibility matrix on the Web site <http://support.peregrine.com>.

AssetCenter databases

This migration supports the conversion of the following databases:

- AssetCenter version 3.01 and later (including the version 4.0.0), and included Service Packs.

If the format of your database is earlier than the version 3.01, you must first convert your database to the 3.02 format.

To learn how to convert a database to the 3.02 format, refer to the following guides:

- **AssetCenter - Version 3.0 - Installing and updating guide**, chapter **Updating AssetCenter**.

- **Readme.txt** of the version 3.02, section **Foreword**.
 - AssetCenter Cable and Circuit 3.10.
-

 **Important:**

The source and target language must be the same during the migration.

Example: You cannot migrate a German version 3.6.0 of AssetCenter to an English version 4.1.0.

Disk space required for the DBMS server

The disk space allotted by the DBMS server to the database to convert must be at least twice the size of the database.

2 Overview

CHAPTER

What does migration entail?

This migration is a set of operations required to convert an earlier version of AssetCenter to the version 4.1.0:

- Converting the working database (structure and contents) in order to make it compatible with the 4.1.0 version of AssetCenter.
- Updating the AssetCenter programs to the version 4.1.0 on all administration and user machines.

Because converting a database is a complex process, this chapter begins by providing you some general principals.

On the other hand, because updating programs is a rather classic manipulation, we will not explain its general principals in this guide.

What does the conversion entail?

Converting a database entails:

- Conforming the structure of the current database to that of the 4.1.0 version of AssetCenter.

- Conserving original data whenever possible.
- Modifying the data that cannot be conserved in their original states due to the change of the database's structure. These modifications are performed automatically whenever possible, and manually otherwise.

What is converted with tools?

- The entirety of the database structure.
- Most of the data.

The data that references the tables, fields and links in the database, however, must be verified and possibly modified manually.

To obtain a list of such data, consult this guide's chapter **References (Migration)**, section **Application data to be manually converted**.

Warning:

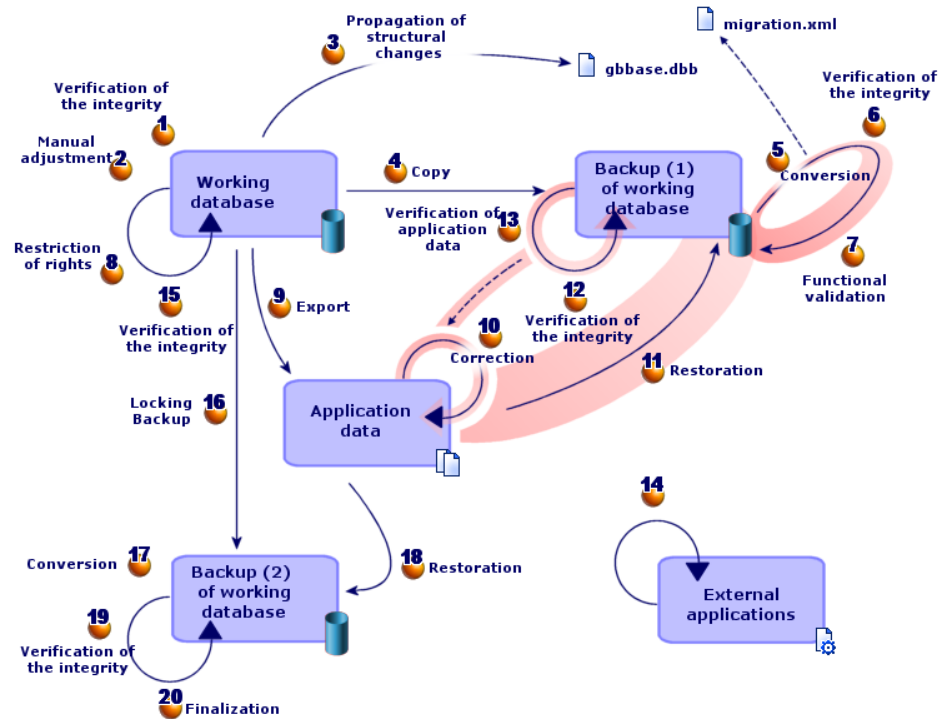
The conversion tools can only be used to modify the structure of the AssetCenter 4.1.0 database when propagating the changes of structure made to the source database for the strict needs of migration.

Peregrine Systems certified technicians may also transfer features to new fields respecting the instructions given in this documentation.

How is the conversion performed?

Here are the main steps in the conversion process:

Figure 2.1. Conversion - process



The conversion is performed in several steps, with or without using additional tools:

- 1 Simulate the conversion on a backup (1) of the working database:
 - 1 Verify the integrity of the working database using AssetCenter Database Administrator.
 - 2 Manually adjust the working database using AssetCenter. This prepares the database in order so that it can be converted.

3 Propagate the structural changes you made to the database to be converted to the standard, database-description file **gbbase.dbb** of the version 4.1.0.

4 Make a backup (1) of the working database. While you simulate the conversion on this backup (1), the users continue to work on the real database.

5 Convert the backup (1) of the working database to the version 4.1.0 using AssetCenter Database Administrator. Adapt and test the **migration.xml** conversion file if necessary.

6 Verify the integrity of the backup (1) of the working database using AssetCenter Database Administrator.

This enables you to make sure the conversion has no corrupted the database.

7 Validate the backup (1) of the working database.

This enables you to make sure the conversion has properly transformed the data as specified.

8 Restrict certain rights to the working database so that the users cannot modify the application data.

9 Export the application data to be manually converted using AssetCenter Database Administrator.

10 Verify the application data to be manually converted using AssetCenter Script Analyzer. Correct any errors.

11 Restore the manually converted application data in the backup (1) of the working database. Do this using AssetCenter Script Analyzer or AssetCenter Database Administrator.

12 Verify the integrity of the backup (1) of the working database using AssetCenter Database Administrator.

This enables you to make sure the restoration has not corrupted the database.

13 Test the restored application data using AssetCenter 4.1.0.

14 Prepare for the adaption of the AssetCenter 4.1.0 integration with external applications.

This will save you time at the end of the conversion.

- 2 Convert the backup (2) of the working database:
 - 15 Verify the integrity of the working database using AssetCenter Database Administrator.
 - 16 Block the working database and make a backup (2).
 - 17 Convert the backup (2) of the working database to the 4.1.0 format using AssetCenter Database Administrator.
 - 18 Restore the manually converted application data in the backup (2) of the working database. Do this using AssetCenter Script Analyzer or AssetCenter Database Administrator.
 - 19 Verify the integrity of the backup (2) of the working database using AssetCenter Database Administrator.
 - 20 Finalize the backup (2) of the working database using AssetCenter to finish the conversion. It is this backup of the working database that you will put into production after upgrading the programs.

How do the conversion tools work?

The conversion tools are integrated into:

- AssetCenter Database Administrator 4.1.0
- AssetCenter Script Analyzer

These programs are launched from the AssetCenter program group.

The tools integrated into AssetCenter Database Administrator are accessible via the following menus:

- **Action/ Diagnostics / Repair database**

This tool verifies and restores the current database.
- **Migration/ Propagate the customized structure**

This tool propagates the customizations you might have made to the database structure you want to convert. These customizations are propagated to the **gbase.dbb** database-description file in the 4.1.0 version.
- **Migration/ Export application data**

This tool exports a copy of the application data to be manually converted in an XML format, which enables you to manually touch it up.

- **Migration/ Convert the database**

This tool converts the structure of the current database according to the specifications of the **migration.xml** conversion file.

- **Migration/ Restore the application data**

This tool imports the analyzed and corrected application data.

How does the conversion process differ from previous versions?

Converting the database no longer involves importing earlier data into an empty database, which was the case before the 4.0.0 version.

The conversion tools perform the necessary modifications directly in the source database.

This new technique provides numerous advantages:

- The duration of the conversion is considerably diminished.
- The data stored in the fields that continue to exist in the new structure are not modified. The duration of the conversion is thus reduced again (because this data does not need to be imported).
- You can customize the **migration.xml** conversion file:
 - The file is in XML format.
 - The file can be edited with a simple text or XML-text editor.
 - The file is largely independent of the DBMS: It is converted in SQL commands for the DBMS.

AssetCenter Script Analyzer enables you to manually convert the exported application data using the **Migration/ Export application data** menu before restoring them.

Why does certain application data need to be manually converted?

Not all data nor all parameters can be automatically converted.

This is especially the case with data and parameters that contain Basic scripts (which sometimes use AssetCenter's AQL querying language): actions, queries, field default values, etc.

To learn which data and parameters need to be manually converted, consult this guide's chapter **References (Migration)**, section **Application data to be manually converted**.

Migrating a database whose DBMS is not supported by version 4.1.0

If the DBMS of the working database is not supported by version 4.1.0:

- 1 Transfer the working database to be converted to a DBMS that is supported by AssetCenter 4.1.0.

To learn how to do this, refer to the **Administering the database** guide, chapter **Creating an AssetCenter database**, section **Changing your DBMS**.

- 2 Proceed to the migration as it is described in this guide.

Limitations of the Procurement module

After having converted the database, you will no longer be able to:

- Receive (receipt) the orders that were partially received before the conversion.
- Return the assets received before the conversion.

We thus recommend that you perform these operations before converting the working database.

Complexity of the migration

The methodology presented in this guide helps you anticipate and avoid numerous problems.

This methodology must be adapted to your company's own manner of using AssetCenter, however.

The complexity of the conversion depends on the degree of customizations made to the database to convert.

3 Step-by-step migration - preparation phase

CHAPTER

This chapter explains step-by-step which operations to perform before working on the database conversion.

Preliminary analysis

Before implementing a migration process, you need to start by doing a complete analysis of your needs and your constraints:

- 1 Make sure you can handle all aspects of the migration as described in this guide.
- 2 Learn about the modifications made to AssetCenter 4.1.0.
To learn about these modifications, refer to the documentations referred in this guide, chapter **References (Migration)**, section **Other documentation (Migration)**.
- 3 Determine what impact these modifications (new functions, modifications of functions, etc.) will have on your use of AssetCenter.

- 4 Determine when you want to implement these new functions (at the same time as the migration or later).
- 5 Update the project specifications (work organization, data organization, parameter configuration, etc.) according to these impacts.
- 6 Update the documentations for users and their training.

Launching the migration project

Taking into account the extent of the improvements and changes made to the version 4.1.0 of AssetCenter, the migration process needs to involve those people in charge of:

- Nomenclature
- Deploying the functional modules:
 - Procurement
 - Contracts
 - Financing
 - Cable and Circuit
- Inventory
- Customizing the database.
- Creating, reports, queries, workflow schemes, actions, etc.
- Integrating AssetCenter with external applications.
- Training users
- Supporting users

It is important to identify and inform these people from the onset of the project.

 **Tip:**

We recommend that you find your project specifications that you used to implement your previous versions.

A project-initialization meeting should take place involving all the people previously mentioned to expose the purpose of the migration, divide its tasks and define its planning.

If your use of AssetCenter is quite advanced (numerous integrity rules, automatic mechanisms, parameter customizations), you can assign teams of people to each functional or technical domain, under the coordination of the project manager.

 **Warning:**

The migration covers several technical aspects. Thus, each team should possess at least one competent engineer. In particular, if you think you might modify the **migration.xml** conversion file that was provided by default, you will need someone with extensive SQL knowledge.

If you want to immediately take advantage of these new functions, you must revise your project specifications and reconfigure your parameters.

 **Tip:**

Do not hesitate to call on Peregrine Systems or its partners, who can provide you specialized and experienced consultants willing to step in at any stage of the migration project.

Training the users and support technicians

When you migrate your programs and convert the AssetCenter database, you might also want to think about training those people who use and support the use of AssetCenter.

To do this:

- 1 Define your training needs.
- 2 Define a training calendar.
- 3 Prepare the training material.
- 4 Update the user notes.

Warning:

Users of AssetCenter need to be trained on the new version before it is put into production.

Tip:

Do not hesitate to call on Peregrine Systems or its partners, who can provide specialized and experienced consultants willing to handle your training needs.

Preparing your conversion computer

Before you can convert your working database, you must prepare a computer adapted for this conversion.

This chapter lists everything you need to install on the conversion computer.

Installing the AssetCenter version corresponding to the database to convert

You need to install this version to access the working database. At the least, you must install the basic module.

Verifying you have access to the working database

You need to do this in order to:

- Prepare the working database for the conversion.
- Make a backup of the working database to prepare for the conversion.
- Convert the working database.

Installing AssetCenter 4.1.0

You need to install at least the following components:

- AssetCenter client
- AssetCenter Database Administrator

- Documentation
- Log viewer
- Migration
- Datakit
- AssetCenter Export

Installing Connect-It (version delivered with AssetCenter 4.1.0)

You need to install Connect-It in order to restore the application data to be manually converted after the last corrections.

Installing an XML file editor

The installation of an XML file editor is optional (a standard text editor is sufficient), but it is quite handy for editing the **migration.xml** conversion file and verifying proper XML structure.

Installing the Sun Java Runtime environment (the version provided with AssetCenter 4.1.0)

You will need this tool to convert the customizations made to the database structure.

Factors conditioning the conversion rate

- DBMS performances
- Throughput between the AssetCenter Database Administrator machine and the database machine.
- Performances of the machines where AssetCenter Database Administrator and the database are installed (but only minimally).

 **Tip:**

If you have a large volume of data in the database to convert, you must position the computers where AssetCenter Database Administrator is installed as close as possible to the database (without going through a WAN network, for example). This is true in particular for tables containing very long fields and binary data (**amComment** and **amImage**, for example).

Preparing the DBMS server

Allotting enough space to the database to convert

During the migration, you will have to convert several backups of the database.

You must make sure you have allotted sufficient space to each database to convert. If this is not done, the conversion risks failure.

To find out how much space you will need, refer to this guide's chapter **Supported environments**, section **Disk space required for the DBMS server**.

Rollback segments

 **Note:**

Rollback segments is the terminology used by Oracle.

Its equivalent in Microsoft SQL Server and Sybase Adaptive Server is **transaction logs**.

All rollback segments must be defined to support the largest required transaction during the conversion.

This transaction consists of performing an `INSERT` in one single operation on the entirety of the table occupying the most space.

4 | Step-by-step migration - simulating the conversion of the working database

CHAPTER

Before you can convert your working database, you must perform simulations of this conversion.

These simulations cannot be performed on the real working database, though. They can only be done on a backup (1).


At the same time, the users can continue to use the working database normally.

After the simulations are complete, you can convert the real working database.

It is this backup (2) that will be put into production.

This chapter explains step-by-step which operations to perform on the backup (1) of the working database.

1 Verify the integrity of the working database

- 1  **Important:**
Make a backup of the working database.
- 2 Launch AssetCenter Database Administrator 4.1.0.
- 3 Connect to the working database (**File/ Open** menu, **Open existing database** option).
- 4 Display the database-diagnostics window (**Action/ Diagnostics / Repair database** menu).
- 5 Select (**All tables**) in the list of tables.
- 6 Specify the name and the location of the log file.
- 7 Select all the verification options.
- 8 Select the **Repair** option.
- 9 Click **Start**.
- 10 Consult the messages of the execution window.
- 11 Consult the log file if necessary.

 **Tip:**

You can use the Log viewer program to consult the log file.

For more information about the analysis and repairs program, consult the **Administering the database** guide, chapter **Diagnostic and repairs of a database**.

2 Manually adjust the working database

 **Warning:**

This section does not apply to users that convert a 4.0.0 database. Those who converted a 4.0.0 database do not need to perform any manual operation in the database before its conversion.

 **Warning:**

Before performing the adjustments described in this section, we strongly recommend that you make a backup copy of your working database.

Certain data must be modified before converting the database in order that the process is carried out smoothly.

Most of the constraints to respect in the database to convert are inferred by the `Mapping` elements of the **migration.xml** conversion file.

This section provides the list of constraints inferred by the standard conversion files. If you modify the standard conversion files, you should identify and verify the constraints inferred by your own changes.

Updating the **amCounter** table

This section concerns users who modified the stored procedure **up_GetCounterVal**. This procedure manages the **amCounter** table according to the directives of the following technical notes:

- Microsoft SQL Server: TN317171736
- Sybase Adaptive Server: TN941931
- Oracle Workgroup Server: TN12516652
- DB2 UDB: TN1029175140

If you made the modifications described in these technical notes, certain records in the **amCounter** table are no longer updated by the stored procedure **up_GetCounterVal**.

Thus, before converting the database, you must:

- 1 Manually update the counters in the **amCounter** table that were diverted to other tables.
- 2 Restore the stored procedure **up_GetCounterVal** to its original state.

 **Tip:**

You will reapply the directives in the technical notes after the final conversion of the working database.

Mandatory nature of fields and links

The conversion will not be started after the initial tests if a mandatory field or link is empty (or becomes empty) during the conversion.

You must make sure that the fields and links that are declared mandatory in the **gbase.dbb** 4.1.0 database-description file are populated before the conversion.

Fields and links whose mandatory nature is declared in an absolute manner or that can become so by applying a script must have an explicate association (described in the **migration.xml** conversion file) or an implicate association (automatically deduced when fields or links share the same SQL name).

If ever you deleted the mandatory nature of a field or link during your use of AssetCenter, it is possible that records were created without this field or link being populated.

In certain cases, these mandatory fields and links need to be populated in the standard structure of the source database in order for it to be converted.

This is the case, for example, with the **lCategId** field in the **amAsset** table.

If you have any doubts about populated links, verify that its external key is populated.

Length of field values

Certain fields of the database to convert are used to populate other fields in the 4.1.0 database.

Certain of these source fields are longer than the destination fields.

In case of problems, you must verify that the length of the values stored in these source fields does not exceed the size of the destination fields.

If this problem comes up, you can solve it by:

- Reducing the length of the source values.
- Increasing the size of the target field (in the 4.1.0 **gbase.dbb** file).

Values that are too long will be truncated during the conversion.

^ character

This character should not be in any of the values of the fields in your working database, and certainly not in any of the values of the following fields (you can determine which of these fields you use in your version of the database):

Table 4.1. Fields that must not contain the ^ character - list

SQL name of the table	SQL name of the field
amProduct	Model
amSoftware	Name
amCatalog	Code
amCompany	Code
amCompany	Name
amProdSupp	PriceCur
amCatProduct	FullName
amAccessRestr	SQLName
amAssetRent	Code
amBrand	BarCode
amBudgClass	Code
amBudgClass	Name
amBudget	Code
amBudget	Name
amBudgetCategory	Code
amCategory	Name
amCategory	BarCode
amCategory	FullName
amCategory	sLvl
amCntrRent	Code
amDateAlarm	Code
amDeprScheme	Code
amEscSchLevel	Code
amFloorPlan	Code
amFuncDomain	SQLName
amFuncDomain	Name
amReservation	ItemNo
amLocation	BarCode
amLocation	FullName
amLocation	Name

SQL name of the table	SQL name of the field
amLossValRule	Code
amModel	BarCode
amModel	FullName
amModel	Name
amContract	Ref
amNature	Code
amNature	Name
amNews	Topic
amPeriod	Name
amPeriod	Code
amEstimate	PONumber
amEstimate	EstimNumber
amPordLine	FullName
amPordLine	ItemNo
amEstimLine	FullName
amEstimLine	ItemNo
amPortfolio	Code
amPortfolio	FullName
amConsUse	ItemNo
amAsset	FullName
amAsset	AssetTag
amProdCompo	FullName
amProfile	SQLName
amProject	Code
amReceipt	ReceiptNumber
amRequest	ReqNumber
amSoftLicCounter	Code
amThirdParty	Code
amUserRight	SQLName
amPOrder	PONumber

Product packages

When the following links are connected:

```
Product P1 -> Package C1 of product P1 -> Product P2
corresponding to the package C1 -> Package C2 of product P2
-> Product P3 corresponding to the package C2
```

- The set Product P1 -> Composition C1 of product P1 -> Product P2 corresponding to the composition C1 is correctly converted.
- The set Product P2 -> Composition C2 of product P2 -> Product P3 corresponding to the composition C2 is correctly converted.
- On the other hand, the nesting of links is interrupted at the level of the link between P2 and C2.

This means that you lose the trace of P1 being composed by P3.

If you want to keep a trace of the link between P3 and P1, you must add a new package C3 to the product P1, and relink P3 to C3.

This must be done before the conversion.

License contracts

License contracts are converted using a process described in this guide, chapter **Step-by-step migration - simulating the conversion of the working database**, section **Convert the backup (1) of the working database/ License contracts**.

If you do not want the license contracts to be processed in this way because you still want them to be contracts:

- 1 Set the **ILicCntrId** field to **0** for all the assets linked to the license contracts that you will leave in the **amContract** table.
- 2 Possibly link these same assets to these same contracts by the **AstCntrDescs** link (which creates records in the **amAstCntrDesc** intermediary table).

Elementary adjustments

Verify that the **lAdjustId** foreign key is not set to **0** for all the records in the **amFieldAdjust** table.

Itemized-list values

Verify that the **Value** field is not NULL for all the records in the **amItemListVal** table.

Procurement and Workflow modules

We recommend that you finish as many running executions as possible before the conversion (partially received orders, items to return, workflows, etc.).

You should especially finish executing workflows concerning license contracts that will be deleted during the conversion.

Note:

The license contracts are the records in the **amContract** table:

- For which the **seType** field is set to 5.
 - That are linked to at least one asset by the **ILicCntrId** foreign key (in the **amAsset** table).
-
-

Warning:

We also recommend that you carefully conserve a copy of the unconverted database as a reference in case you run into any problems during the conversion.

Product suppliers

The **amProdSupp** table is no longer available in version 4.0.0 and later. During conversion, the records from the **amProdSupp** table are transferred to the **amCatRef** table if the currency in which the **mPrice** field (from the **amProdSupp** table) is declared in one of the following ways in the **amCurrency** table:

- Default currency
- Reference currency 1
- Reference currency 2

The records of the **amProdSupp** table that do not meet these conditions are not converted.

If you need to manage other currencies, you can do one of the following:

- Convert the **mPrice** field to an adequate currency before converting the database.

 **Tip:**

You can obtain a Euro currency converter from Peregrine Systems technical support.

-
- Reassign the other currencies to the following items:

- Default currency
- Reference currency 1
- Reference currency 2

if the currently assigned currencies are not used in the database.

- Add `Mapping` elements to the **migration.xml** file for each additional currency to process.

To learn more about this solution, refer to this guide's chapter **Adapting the migration.xml conversion file**.

A `Mapping` type element is proposed in the **migration.xml** conversion files.

To find it, you must open the conversion file and search for the text: Use the following mapping to add another currency.

With the default **migration.xml** files, the conversion tool creates up to 3 records per supplier in the **amCatalog** table (1 for each supported currency).

The **amCatRef** table references are associated with one of these catalogs during conversion.

Estimate

During the conversion, the records from the **amEstimate** table are transferred to the **amPOrder** table. The **seStatus** field is set to **Quoted**.

Any estimate containing an estimate line whose **IPordLineId** field is not set to **0** is deleted during the conversion. (We consider the estimate to have been transformed into an order, which will be converted. This corresponds to how AssetCenter) 4.1.0 manages estimates.)

You can take advantage of this opportunity before converting to delete all useless estimates from the **amEstimate** table before the conversion. This assures that you do not uselessly overload the **amPOrder** table.

If you still want to conserve these estimates, however, you can set the **IPordLineId** field to **0** for all estimate lines to conserve during the conversion.

Products packages

For a clean conversion, the tree structures of product compositions (**amProdCompo** table) must have at least 9 levels.

To respect this condition, reorganize the product packages whose **sLvl** field is superior or equal to **9**.

Furthermore, in the case where a record from the **amProdCompo** table is linked to:

- A main product (**MainProduct** link whose **bSuppPackage** field is set to **1**) ...
- An asset by the **UsedAsset** link or a contract by the **UsedContract** link,

Then the **UsedAsset** or **UsedContract** link is not transferred during the conversion.

If you want to transfer these links, you must set the value of the **bSuppPackage** field of the main product to **0**.

Order lines

For a clean conversion, the tree structures of the purchase orders (**amPordLine** table) must have at least 10 levels.

To respect this condition, reorganize the order lines whose **sLvl** field is superior or equal to **10**.

Categories

For a clean conversion, the tree structures of categories (**amCategory** table) must have at least 10 levels.

To respect this condition, reorganize the categories whose **sLvl** field is superior or equal to **10**.

Order lines

For a clean conversion, the tree structures of the order lines (**amOrdLine** table) must have at least 10 levels.

To respect this condition, reorganize the order lines whose **sLvl** field is superior or equal to **10**.

Budgets

In the default **migration.xml** conversion files, the contents of the **amBudget** table are transferred to the **amCostCategory** table.

This behavior is suited if you have been using budgets for cost accounting purposes (as cost centers) and not to manage budgets in the truer sense of the term.

If you have been using budgets as budgets (and not as cost centers), you must adapt the **migration.xml** conversion file so that such budgets are transferred to the **amBudgLine** table.

For this purpose, inactive **Mapping** elements were inserted into the **migration.xml** files to provide you with the basis of an association between **amBudget** and **Budget lines**.

If you activate these **Mapping** elements during the conversion:

- The budgets (**amBudget** table) are processed differently depending on whether the **dStart** and **dEnd** fields are populated or not.
 - If even one of these 2 fields is not populated, the conversion program only moves the records to the **Cost categories** table (**amCostCategory**).

- If these 2 fields are populated, the conversion program moves the records to the **Budget lines** table (amBudgLine) and the **Cost categories** table.
- You must thus make sure that the **dStart** and **dEnd** fields are populated, according to the result you want to obtain during conversion.

Propagate database-structure changes

Warning:

To perform this operation, **gbbase.dbb** the 4.1.0 database description file that you reference here must be the standard file installed with AssetCenter 4.1.0, without modifications.

This operation:

- Concerns the users who modified the standard structure of the database to be converted and want to keep those changes in the new database.
- Aims to propagate the structural modifications in the standard **gbbase.dbb** file of AssetCenter 4.1.0.

Tip:

The **gbbase.dbb** obtained will be used to create the structure of the target database during the conversion.

- Uses a tool dedicated to this operation, which is available in AssetCenter Database Administrator.

Warning:

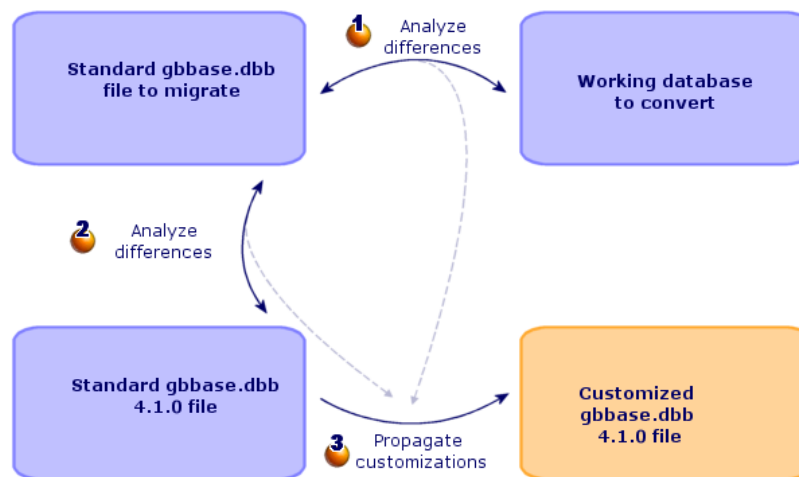
Only the structural changes made using AssetCenter Database Administrator will be accounted for.

You must manually delete all structural changes made by any other means in the database to be converted.

General overview

The following describes the process of propagating structural changes:

Figure 4.1. Propagating structural changes - process



1 The tool determines the differences between the structure of the database to convert and the standard structure of the new version.

2 The tool determines the differences between the standard structure of the version to migrate and the standard structure of the version 4.1.0.

3 The tool copies and modifies the standard **gbbase.dbb** file of the version 4.1.0 according to what it identified during steps 1 and 2. It does so by respecting the following rules:

- The modifications performed on the standard tables that disappear in the version 4.1.0 are lost.
- If a modification is detected for the same table, field or link in the steps 1 and 2, it is the modification detected at step 2 that is applied. A warning message is displayed.

Direct consequence: In the source database to convert - before definitively propagating the structural changes - you must modify the SQL names of tables, fields and indexes that appear in the version 4.1.0 but whose use is different from that in the database to convert.

The 4.1.0 **gbbase.dbb** file with the propagation of structural changes must be clearly identified. This will come in handy in the following steps:

- Export the application data to be manually converted.
- Correct the application data to be manually converted.
- Convert the backup (1) of the working database.
- Convert the backup (2) of the working database.

Propagate the structural changes.

- 1 Launch AssetCenter Database Administrator 4.1.0.
- 2 Connect to the backup (1) of the working database with the **Admin** login (**File/ Open/ Open existing database** menu).
- 3 Select the **Migration/ Propagate the customized structure** menu.
- 4 Populate the **Generation, folder** field: This is an empty folder of your choice in which will be saved to 4.1.0 **gbbase.dbb** file with structural change propagation.
- 5 Click **Build**.
- 6 Consult the messages that appear on screen.

Consult the **newdbb.log** log file, which is located in the folder defined by the **Generation** folder field.

 **Tip:**

You can use the Log viewer program to consult the log file.

- 7 If the messages tell you so, modify the unconverted database structure. Then perform the migration starting from the step **Copy (1) the working database**.

This must be repeated until you obtain a good **gbbase.dbb** file without any problem messages.

- 8 Certain scripts might not be propagated to the **gbbase.dbb** 4.1.0 file. There will be a message in the **newdbb.log** log file and an **.xml** file created in the **<Generation folder>\dbbscript** and **<Generation folder>\bulddb\dbbscripts** folders for each script that is not propagated.

These customizations must be propagated manually in the **gbbase.dbb** file obtained using the **Migration/ Propage the customized structure** menu item.

You can go to step **Verify and correct the application data to be manually converted** to perform this operation, if you want to AssetCenter Script Analyzer to convert the scripts.

AssetCenter Script Analyzer will suggest modifications to be made, which you perform manually in the **gbbase.dbb** 4.1.0 file using AssetCenter Database Administrator.

- 9 If you are converting a 4.0.0 database, verify using AssetCenter Database Administrator that each page you added is still valid. If this is not the case, you must correct each one manually.
-

 **Warning:**

However, you will need to modify the customized 4.1.0 **gbbase.dbb** file again when you execute the step **Convert the backup (1) of the working database**.

Potential conflicts

If the propagation of structural changes is abnormally interrupted, verify if there is an **xerces.jar** file in the **/jre/lib/ext** sub-folder of the Java installation folder.

If there is, temporarily move this folder and try to execute the propagation of structural changes again.

Analyzing and adapting the migration.xml conversion file to handle structural changes

If the structural changes that were propagated include table additions, you must modify the **migration.xml** conversion file so it manages the conversion of these tables.

4 Copy (1) the working database

Problems that can occur during a traditional backup

If you make a backup of the working database using DBMS tools, the backup will be identical to the original for everything concerning additions, modifications or deletions of the following events using tools other than AssetCenter Database Administrator:

- Index
- Triggers
- Stored procedures
- Views

However, the conversion program cannot manage these structural modifications.

You must delete these structural modifications before converting the database.

We propose two methods for making a backup that conform to the conversion's requirements:

- Make a backup using the DBMS tools, and cancel the structural modifications listed in this section.
 - Perform a dump of the existing database, then restore it in an empty database.
-

 **Note:**

The backup of this database must be accessible via the conversion computer. To learn how to make a backup of your database, consult the DBMS documentation.

Solution 1: Copy the database using the DBMS tools

- 1 Copy the database using the DBMS tools.
The backup of this database will be identical to the original database.
- 2 Delete all the modifications made to:
 - Indexes
 - Triggers
 - Stored procedures
 - Views

Solution 2: Dump/restore the database

- 1 Perform a dump of the working database using AssetCenter Database Administrator.
- 2 Create an empty database.
- 3 Restore the dump in the empty database.

This method is advantageous for deleting all modifications made to the items listed above.

To learn how to perform a dump/restore of a database, refer to the **Administering the database** guide, chapter **Creating an AssetCenter database**, section **Changing your DBMS**.

After having copied the working database

Create an AssetCenter connection to the backup (1) of the working database.

5 Convert the backup (1) of the working database

⚠ Warning:

The conversion tools must not be used to modify the structure of the database 4.1.0 (adding, deleting or modifying tables, fields, indexes, stored procedures, triggers, views, etc.).

Such modifications must be planned after the migration.

Adapting the migration.xml conversion file

⚠ Warning:

This operation must be carried out by a Peregrine Systems certified technician for the migration.

Peregrine Systems declines all responsibility if this condition is not strictly adhered to.

AssetCenter 4.1.0 is installed with conversion files by default (1 file per version of AssetCenter that is supported by the migration).

These files describe what data to transform during the database conversion and what transformations to perform.

The conversion files are called **migration.xml**.

They are generally located in the **C:\Program Files\Peregrine\AssetCenter\migration\fromxxx** folder, where **xxx** is the number of the earlier version.

If you use AssetCenter in a standard manner, you can probably use one of the conversion files installed by default.

If you have particular needs (features to transfer to the fields of the new database, fields using different functions other than their default functions, added tables and fields, etc.) you must adapt the conversion file to your needs.

 **Warning:**

The standard or customized conversion file must be tested on a backup (1) of the working database before being definitively executed on the working database.

 **Important:**

When you customize the **migration.xml** conversion file, you must neither rename it nor replace it. This is because the tools that use this file will search for it in the standard folder.

We also recommend that you make a backup of this conversion file before starting to modify it.

To learn about the syntax of the conversion files and how to customize them, refer to this guide's chapter **References (Migration)**, section **Adapting the migration.xml conversion file**.

Converting the backup (1) of the database

To convert the database:

- 1 Launch AssetCenter Database Administrator.
 - 2 Connect to the backup (1) of the working database to convert with the **Admin** login (**File/ Open/ Open existing database** menu).
-

 **Important:**

In the connection detail of the level of AssetCenter:

- The **Owner** field must not be populated.
 - The **User** field must reference a user that is the **owner** of the database tables (creation rights for all database objects).
-

- 3 Select **Migration/ Convert the database** menu.

- 4 Populate the **Dest. database description** field: full path of the 4.1.0 **gbase.dbb** file with propagation of structural changes. (This file is located in the folder you selected using the **Migration/ Propagate the customized structure, Generation folder** field.)
 - 5 Populate the **License file** field: full path of the AssetCenter 4.1.0 license file, **license.cfg**. (This file is provided with AssetCenter 4.1.0.)
-

 **Note:**

Starting with the version 4.0.0 of AssetCenter, access to the database is controlled by a license file.

To learn more about activating a license file, consult the **Administering the database** guide, chapter **Creating an AssetCenter database**, section **Selecting a license file**.

- 6 Populate the **Migration file** field: full path of the conversion file corresponding to the version of the database to convert (usually **C:\Program Files\Peregrine\AssetCenter\migration\fromxxx\migration.xml**).
 - 7 Populate the **Log folder** field: folder where the **sdu.log** log file will be saved (example: **c:\temp**).
 - 8 Click **Update**.
 - 9 Consult the messages that appear on screen.
 - 10 Consult the messages of the **sdu.log** log file.
-

 **Warning:**

If even a minor error occurs during the conversion, you must:

- 1 Correct the source of the problem.
 - 2 Restart the conversion from step **Copy (1) the working database**.
-

Information about the conversion

Here are some rules that are used during the conversion.

 **Tip:**

If you want to obtain a different behavior, modify the corresponding associations in the **migration.xml** conversion file.

Rules used for all source versions of the database

Floor plan positions

Records in the **amFloorPlanPos** table are deleted:

Structural parameters of the database

The conversion program applies to all the parameters of the tables, fields, links and indexes defined in the selected 4.1.0 **gbbase.dbb** database-description file.

This is the case, for example, of the script that calculates the default value of fields.

Mandatory fields

If a destination field:

- Is mandatory or if it is part of an index requiring unique values.
- And it is not a part of an explicit association (described in the **migration.xml** conversion file) or an implicit association (automatically deduced when fields share the same SQL name).

Then a warning message will appear in the first phase of conversion.

This is the test phase that precedes any modification to the database.

The conversion is not interrupted unless you provoke this interruption yourself.

If you decide to interrupt the conversion, you must do so before any modifications have been made. Otherwise, you will have to restore the backup (1) of the unconverted working database.

You might want to populate the information necessary in order for the mandatory fields be populated. This information should go into the unconverted working database.

Default values of fields

The default values defined in the structure of the working database are not applied.

If you want an equivalent of the default value to be applied, you must define this in the conversion file.



The **migration.xml** standard conversion files already contain `value` attributes that perform such a task.

Index of unique values

The conversion file does not verify that unique values have been respected.

On the other hand, the DBMS will interrupt the conversion if an operation tries to undermine the integrity of the index.

SQL validity of `value` attributes

The conversion file does not verify the SQL validity of `value` attributes, either.

On the other hand, the DBMS will interrupt the conversion if a `value` attribute that is non-valid in SQL terms is found.

Grouped nature of the conversion

The conversion operations are performed in a **grouped** manner for nearly all data, and not record-by-record, (a global SQL order modifies the records of one whole table).

Modified tables

For one table modified (table **A** in our example), the conversion tool proceeds in the following order:

- 1 Table **A** is renamed (**AOld** in our example).
- 2 A new table is created (**A** in our example).
- 3 The data is transferred from **AOld** to **A**.
A `Mapping` element can define another behavior.
- 4 **AOld** is deleted.

Thus for a given table **A**:

Does table A exist in the old version?	Does table A exist in version 4.1.0?	Are there modifications to fields, links or indexes between the old version and version 4.1.0?	Then the conversion program:
Yes	Yes	No	Works directly on table A .
Yes	Yes	Yes	Creates the intermediary AOld table.
No	Yes	Does not apply	Creates the new table A .
Yes	No	Does not apply	Transfers the data from table A to other tables and deletes the table A at the end of the conversion.

Tip:

The `From` attribute does not need to reference the **AOld** table (referencing **A** is enough; the conversion program knows when to look for information in **AOld**).

On the other hand, in the scripts executed outside of `Mapping` elements, you must distinguish between **A** and **AOld**.

 **Note:**

The unchanged and deleted tables are not renamed during the conversion.

Fields storing application data to be converted manually

The fields that store application data to be manually converted are emptied. (This is what is intended by the **migration.xml** conversion file installed by default.)

Rules used for the versions earlier than the version 4.0.0

History

The records in the **amHistory** table are converted. The information contains the history of the modifications to the contracts when they belonged to unconverted database.

Assets

The following fields are transferred as-is from **amAsset** to **amComputer**:

- ComputerDesc
- BIOSSource
- BIOSAssetTag
- dtBIOS
- lCPUNumber
- SoundCard
- VideoCard
- OSServiceLevel
- OSBuildNumber

If the database to convert is version 3.5.0 or earlier, and if a feature containing an information of the same nature is associated with the transferred and if this feature is populated, then the value of the feature overrides the value obtained by the transfer of the field.

The features have the following SQL names:

- **BiosMachine** (equivalent to the **ComputerDesc** field)
 - **BiosSource** (equivalent to the **BIOSSource** field)
 - **BiosAssetTagId** (equivalent to the **BIOSAssetTag** field)
 - **BiosDate** (equivalent to the **dtBIOS** field)
 - **ICPUCount** (equivalent to the **ICPUNumber** field)
 - **SoundCardDescription** (equivalent to the **SoundCard** field)
 - **GCard01Description** (equivalent to the **VideoCard** field)
 - **OS01ServiceLevel** (equivalent to the **OSServiceLevel** field)
 - **OS01BuildNumber** (equivalent to the **OSBuildNumber** field)
-

 **Tip:**

This task is performed within the <PreActions> element of the **migration.xml** file.

This task is disabled in the **migration.xml** files of version 3.6.0 and later.

If this is useful to you, you can enable the following lines in the **migration.xml** file.

Adjustments

During the conversion, the records in the **amAdjustment** table are transferred to the **amPortfolio** table.

In order not to overload the database, the following fields from the **amAdjustment** table are lost:

- Name
- mTax*
- seAcquMethod
- lReqLineId
- lPOrdLineId
- lDelivLineId
- lInvLineId

Furthermore, the adjustments of license contracts are deleted.

 **Tip:**

If you want to modify these behaviors, you must add the corresponding associations to the **migration.xml** conversion file.

Consumptions

During the conversion, the records from the **amConsUse** table are transferred to the **amPortfolio** table.

At this time, the **mTax*** fields from the **amConsUse** table are lost:

 **Tip:**

If you want to conserve the information stored in these fields, you must add the corresponding associations to the **migration.xml** conversion file.

Product packages

During the conversion, the records in the **amProdCompo** table are transferred in the following manner:

- Those records corresponding to standard configurations (those whose **bSuppPackage** option is set to **0**) are transferred to the **amReqLine** table.
- Those which correspond to supplier packages (those whose **bSuppPackage** option is set to **1**) are transferred to the **amCatProduct** table.

For those records that are transferred to the **amProdCompo** table, the value of the **bInstantAssign** field is set to **1**.

Products

All products (**amProduct** table) are transferred to the **amModel** table. They are also transferred to the **amCatProduct** table if the one of the following conditions is fulfilled:

- The **mPrice** field of the product is different than **0**

- The product is linked to a record in the **amProdSupp**, **amOrdLine**, **amDelivLine** or **amInvoiceLine** table.

When 2 products P1 and P2 are created in the **amCatProduct** table, P2 is a component of P1, and P1 and P2 are both transferred to the **amPortfolio** table, then the **bPreinstalled** field of the records in the **amCatProduct** table is set to 1.

The products are also linked to the **amCatRef** table if the products are linked to a record in the **amProdSupp**, **amOrdLine**, **amDelivLine** or **amInvoiceLine** table.

Installation to create

The records in the **amProdSoftInfo** table establish a link between the license products (**amProduct**) and the software products (**amSoftware**). Their conversion gives rise to the creation of records in the following tables:

- **amCatProduct** (this corresponds to supplier packages).
- **amReqLine** (this corresponds to standard configurations).

License contracts

Warning:

Converting license contracts is a tricky part of the conversion process. This is due to its complexity.

The best way to test your database is to simulate the conversion in a standard fashion, then to verify the result in detail.

The license contracts are the records in the **amContract** table:

- For which the **seType** field is set to 5.
- That are linked to at least one asset by the **lLicCntrId** foreign key (in the **amAsset** table).

Such contracts are converted according to the simplified explanation that follows:

- They are converted into software licenses. For this, they are transferred to the **amPortfolio** table and linked to a model that is, itself, linked to a nature. This nature's **bSoftLicense** field is set to **1**.
- The records in the **amWfInstance** table linked to these contracts are deleted.
The records linked to these deleted workflow instances are also deleted.
- The fields and links specific to the contracts, but which are not relevant to the **amPortfolio** table, are lost.
- The **lSoftLicUseRights** foreign key of the assets linked to these contracts is set to **0**.
- The **seAcquMethod** field is set to **0**.
- The links between the contracts and the assets (stored in the **amAstCntrDesc** table via the **AstCntrDescs** link) are transformed into software installations on these same assets (**amPortfolio**).
- The links between the contracts and the employees (stored in the **amAstCntrDesc** table via the **AstCntrDescs** link) are transformed into user accounts. These user accounts are sub-licenses of the license created in the **amPortfolio** table.
- The records in the **amAdjustment** that were linked to the contracts are deleted.
- The hierarchic link of these contracts is lost.

Potential sources of conflict

- If the maximum number of primary IDs supported by the conversion is reached, you will receive an error message.
If this happens, contact Peregrine Systems user support.

6 Verify the integrity of the backup (1) of the working database

Verify the integrity of the backup (1) of the working database as indicated in this guide's chapter **Step-by-step migration - simulating the conversion of the working database**, section **Verify the integrity of the working database**.

Instead of connecting to the working database, you must connect to the backup (1) of the working database.

Select the option **Analyze only** instead of the **Repair** option.

If problems are displayed by the program, perform one of the following operations:

- 1 Modify the **migration.xml** conversion file.
- 2 Start again from step **Convert the backup (1) of the working database**.
- Or:
 - 1 Modify the data in the working database.
 - 2 Start again from step **Copy (1) the working database**.

7 Validate the backup (1) of the converted working database.

Browse the backup (1) of the converted production database to see if the conversion appears correct.

If you find any anomalies, perform one of the following operations:

- 1 Modify the **migration.xml** conversion file.
- 2 Start again from step **Convert the backup (1) of the working database**.
- Or:
 - 1 Modify the data in the working database.

- 2 Start again from step **Copy (1) the working database.**

Restriction of certain rights on the working database

Modify the user rights of the working database so that users can no longer modify the tables containing application data to be manually converted.

- 1 Determine the list of application data to be manually converted by referring to this guide's chapter **References (Migration)**, section **Application data to be manually converted.**
- 2 Display the list of user rights via the **Administration/ Users rights** menu.
- 3 Select each user right and, for each one:
 - 1 Select all the objects described by the users rights.
 - 2 Unselect the **Create, Delete and Enter during creation** rights.
 - 3 Click **Modify.**

You need to do this because the application data to be manually converted is extracted from the backup of the working database. The modifications made to the backup of the working database are not recovered in the conversion process.

Export application data to be manually converted

Reminder

To learn exactly what **application data** is, refer to this guide's chapter **Application data to be manually converted.**

Exporting the application data to be manually converted

- 1 Launch AssetCenter Database Administrator 4.1.0.
- 2 Connect to the working database with the **Admin** login (**File/ Open/ Open existing database** menu).

- 3 Select the **Migration/ Export application data** menu.
- 4 Populate the **Dest. database description** field: full path of the 4.1.0 **gbbase.dbb** file with propagation of structural changes. (This file is located in the folder you selected using the **Migration/ Propagate the customized structure, Generation folder** field.)
- 5 Populate the **Migration folder** field: folder containing reference files necessary for the conversion.
There is one folder per database version that can be converted (usually **C:\Program Files\Peregrine\AssetCenter\migration\fromxxx**, where **xxx** is the number of the earlier version).
- 6 Populate the **Working folder** field: folder of your choice designed to store the exported application data to be manually converted.
- 7 Consult the messages that appear on screen.
- 8 Consult the **sduxprt.log** log file. This file is located in the folder defined by the **Working folder** field.

 **Tip:**

You can use the Log viewer program to consult the log file.

- 9 Make a copy of the tree structure of the **.xml** files created and save it.
This will come in handy if you want to use the original **.xml** file at a later point in time or to view the modifications that you made to the **.xml** files.

Rules to respect during the export

The export tool:

- Exports a copy of the application data to be manually converted in a format that enables you to manually touch it up.
- Exports, not only the application data to be converted, but also the information about the context of this data. This enables you to update this data easier with AssetCenter Script Analyzer.

- Creates a tree structure of **.xml** files organized by type of application data.

Each **.xml** file corresponds to a record that contains one or more types of application data to verify.

- Includes all the application data that you added yourself to the database.
- Excludes the application data from line-of-business data and sample data that you have not modified.



Tip:

You will import the 4.1.0 version of this data later in the conversion process.

-
- Do not verify if the tables, links and fields of the application data conform to the structure of the 4.1.0 database.



Tip:

This is done by the AssetCenter Script Analyzer.

Processing application data to be manually converted

The application data to be manually converted is processed in several steps:

- 1 Verify and correct the application data to be manually converted
- 2 Restore corrected application data
- 3 Verify the integrity of the backup (1) of the working database
- 4 Verify restored application data

These steps are described in this section.



Note:

In this section, **fields** to be verified and possibly replaced with new values refer to fields and links in the structure of the AssetCenter database.

7 Verify and correct the application data to be manually converted

This step is performed with AssetCenter Script Analyzer.

Verifying and correcting application data

Here are the steps to perform. To learn more about each of these steps, refer to the information included about the graphical layout of AssetCenter Script Analyzer (afterwards).

- 1 Launch AssetCenter Script Analyzer.
- 2 Populate the **Working folder** field.
See 2.
- 3 If you have created a tree structure of **.xml** containing scripts that are not propagated in the step **Propagate database-structure changes** :
 - 1 Copy the **<Generation folder>\dbbscript** and **<Generation folder>\builddb\dbbscript** folders created in the step **Propagate database-structure changes** (if they exist).
 - 2 Paste this folder in the folder specified by the **Working folder** field.
- 4 Display the list of application data to verify (**Actions/List all files** or **Actions/List unprocessed files** menu).
The **Message** window displays the list of **.xml** files to verify against the synthesis data.
See 8.
- 5 If you have planned at this stage to process the scripts that have not been propagated automatically in the step **Propagate database-structure changes**, start with the **.xml** corresponding to these scripts:
 - 1 Select the first **.xml** file from the **<Generation folder>\dbbscript** and **<Generation folder>\builddb\dbbscript** folders.

- 2 Analyze the file in detail (**Actions/List the problems in the script menu**).
 - 3 Consult the **Message** window.
See [13](#) and [14](#).
 - 4 Use the suggestions for modification made by AssetCenter Script Analyzer to modify the corresponding scripts in the **gbase.dbb** file obtained in the step **Propagate database-structure changes**.
For this, launch AssetCenter Database Administrator, open the **gbase.dbb** file and perform the script modifications by hand.
 - 5 When you have finished processing the **.xml** file, select the **Functionally valid** option.
 - 6 Display the list of application data to verify (**Actions/ List unprocessed files** menu).
The Report window displays the list of **.xml** files.
 - 7 Select the next **.xml** file to validate and perform a detailed analysis of this file.
- 6 Select each of the other **.xml** files to verify in the report.

For each **.xml** file selected:

- 1 Analyze the file in detail (**Actions/List the problems in the script menu**).

 **Tip:**

You can have several types of application data to manually convert in the same **.xml** file.

- 2 Consult the **Message** window.
See [13](#) and [14](#).
- 3 Modify the **.xml** directly in the edit zone: the **Context** field and the tabs.
The modified **.xml** file will be imported later in the conversion process.
See [6](#).

- 4 Test the script in its context (**Actions/ Validate the context of the script** menu).

The purpose of this operation is to verify that the script is valid according to the version 4.1.0 database structure.

 **Important:**

This operation is critical for action scripts and SQL queries, because they cannot be opened with a graphical interface in AssetCenter unless they are valid. It thus becomes quite complicated - even impossible - to correct them after having restored the **.xml** files.

This operation verifies that the fields and links between brackets are valid according to the context of the action.

 **Note:**

The script will be automatically tested in its context when you select the option **Restorable** for the current file.


- 5 When you have finished analyzing and correcting the **.xml** file, select the option **Restorable**.

This means that you can restore the **.xml** file in the backup (1) of the working database to test the application data that was manually converted.

See .

- 6 Display the list of application data to verify (**Actions/ List unprocessed files** menu).

The Report window displays the list of **.xml** files.

The **.xml** files marked **Restorable** are no longer analyzed by the AssetCenter Script Analyzer. The number in parentheses is set to **0**. The blue  icon indicates that it is restorable.

- 7 Select the next **.xml** file to validate and perform a detailed analysis of this file.

AssetCenter Script Analyzer menus

Table 4.2. AssetCenter Script Analyzer - menus

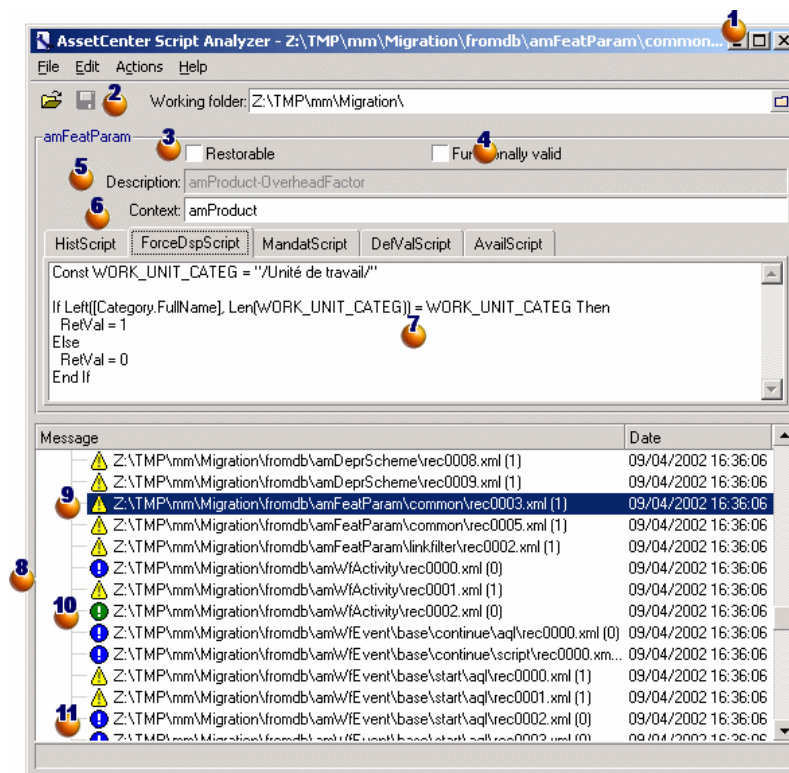
Menu	Use
File menu	
New	Not needed.
Open	Enables you to open an .xml file from the tree structure whose root is defined by the Working folder field.
Save	Saves the modifications made to the file (Restorable or Functionally valid character, context, scripts).
Save as	Not needed.
Exit	Exits AssetCenter Script Analyzer.
Edit menu	
Functions just like any other Edit menu.	
Actions menu	
Open the next file	Opens the next .xml file in the list displayed by the Message window.
Open the previous file	Opens the previous .xml file in the list displayed by the Message window.
List the problems in the script	Analyzes the potential problems of the selected .xml file and displays the result in the Message window.
Validate the script in its context	Tests the validity of the current script according to the table in the Context field (if it is populated). Otherwise, it tests the validity of the script outside the context.
Force the restorable nature of the file	Selects the option Restorable , even if the script is not validated in its context by the Actions/ Validate the script in its context menu.
<p>Warning:</p> <p>Only use this menu when:</p> <ul style="list-style-type: none"> • Using the menu Actions/ Validate the script in its context returns an unjustified error. • You are certain of the script's validity. 	
List unprocessed files	Displays the list of .xml files: <ul style="list-style-type: none"> • From the tree structure whose root is defined by the Working folder field. • Whose Functionally valid option is not selected.
List all files	Displays the list of all .xml files from the tree structure whose root is defined by the Working folder field.


Menu	Use
Restore application data	Enables you to select a connection to an AssetCenter database and import .xml files whose Restorable option is selected. This menu performs the same job as the Migration/ Restore application data menu in AssetCenter Database Administrator.

List of .xml files displayed by AssetCenter Script Analyzer

When you use the **Actions/ List all files** or **Actions/ List unprocessed files** menu, the window displayed by AssetCenter Script Analyzer looks like this:

Figure 4.2. AssetCenter Script Analyzer - *.xml file analysis window



 Full path of the current **.xml** file.

2 Folder containing application data exported with AssetCenter Database Administrator (tree structure of **.xml** files that contain the application data to be manually converted).

This is the file that you specified in the **Working folder** field, via the AssetCenter Database Administrator menu **Migration/ Export application data**.

This is also the folder at the root of which is located the **modifications.xml** file.

This file is generated from the **migration.xml** conversion file.

It describes all the migration possibilities available for source database fields (in order).

The **modifications.xml** file is only used by AssetCenter Script Analyzer to diagnose problems on field names.

3 When you have finished analyzing and correcting the **.xml** file, select the option **Restorable**.

4 When you have finished testing the application data from the **.xml** file that are restored in the AssetCenter database, select the option **Functionally valid**.


5 Information that helps you identify the application data to verify. This information varies (SQL name of the record that stores the application data, for example) and can be extracted during the export of application data with AssetCenter Database Administrator.

6 Context table of the application data (when this context exists).

 **Warning:**


The **Actions/ List the problems in the script** menu does not test this information. You must verify that the context is always valid yourself (deleted table in the version 4.1.0, for example).


7 If the file contains several scripts, each script is in a different tab. If one of the scripts has a problem (field in the **modifications.xml** file), a message is displayed by the **Actions/ List the problems in the script** menu.

 List the **.xml** files of the tree structure whose root is defined by the **Working folder** field. According to the menu used, this list contains all the files (**Actions/ List all files** menu) or only those files whose **Functionally valid** option is not selected (**Actions/ List unprocessed files** menu).

 Each line of this list corresponds to an **.xml** file.



The number in parentheses corresponds to the number of lines of the **.xml** file that contain fields, tables or links to verify.



If the number is **0**, and the line begins with , this does not signal a problem at the level of the SQL names (of fields). It does, however, signal that the file contains incorrect application data in the context of the table defining it (it is probably an incorrect link).

If the number is **0**, and the line begins with , this does not signal a problem at the level of the SQL names (of fields), nor does it signal that the file contains incorrect application data in the context of the table defining it. The file can be restored and tested in the AssetCenter database.

 **Note:**

Click the file once to open it.

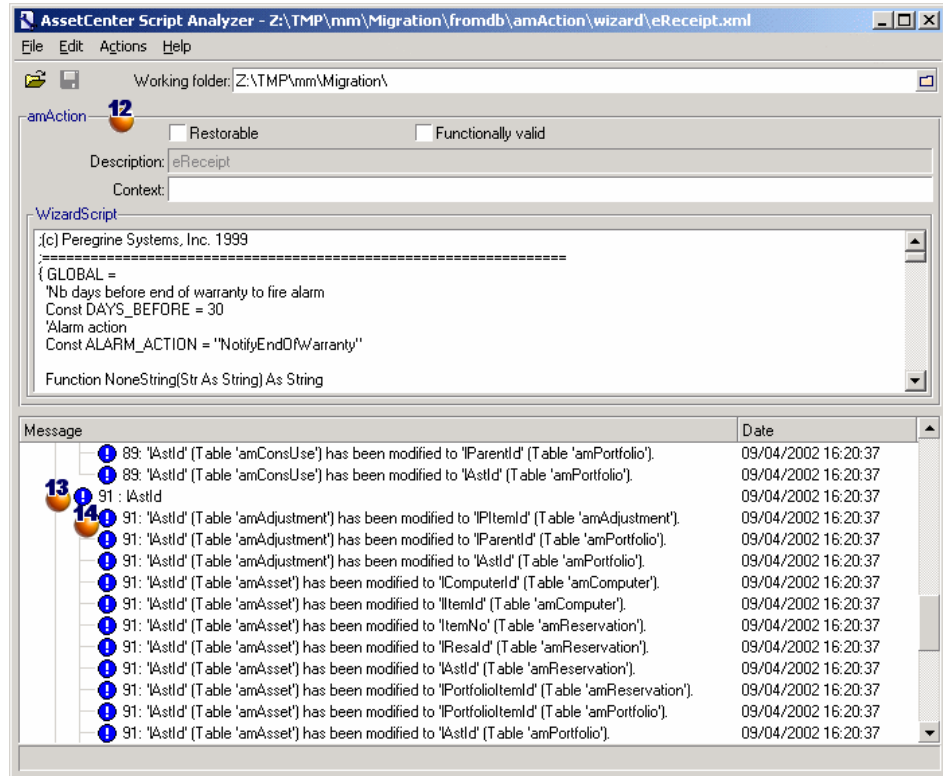
 The green  icon indicates that the **.xml** file is **Functionally valid**.

 The blue  icon indicates that the **.xml** file is **Restorable**. This status is either manually selected by you or automatically when using the **Actions/ List all files** and **Actions/ List unprocessed files** menus. (This is only if none of the **.xml** files are in the **modifications.xml** file and if the script has been validated in its context.)

List the problems in the script

When you use the **Actions/ List the problems in the script** menu, the window displayed by AssetCenter Script Analyzer looks like this:

Figure 4.3. AssetCenter Script Analyzer - script analysis window



12 SQL name of the table at the origin of the application data from the **.xml** file.

13 Line number of the problematic script, followed by the SQL name of the field that was found in the **modifications.xml** file.

To verify: fields and tables whose SQL name is found in the **modifications.xml** conversion file.


The analysis program does not take into account the table in which the fields and links belong. The **modifications.xml** file can be considered questionable even if one field's SQL name appears in it.

It could be that the SQL name is both an unchanged field in a table and the name of a field modified in another table. This is what the program helps you determine and possibly correct.

During the search for SQL names of tables, fields and links in the **modifications.xml** file, the following are considered to be delimiters: all alpha-numeric characters except the character `_`.

 **Note:**

Double-clicking the mouse places the cursor on the problematic line.

 Each sub-line corresponds to a modification proposition.

This window displays one line per possible correction for an SQL name of the field to verify.

The number at the head of the line corresponds to the number of the line to verify in the **.xml** file.

Each proposed correction comes from one of the associations described in the **modifications.xml** file.

The propositions are a result of the associations found in the **modifications.xml** file.

There are several types of messages:

- 'A' (Table 'B') was modified in 'C' (Table 'D'):
The A field of the script is part of table B in the source database. The A field was associated in the **modifications.xml** file with the C field, which is part of the table D in the **gbase.dbb** destination structure.
Example: 'script' ('amAction' table) was modified in 'memScript' ('amAction' table)
- 'A' (Table 'B') no longer exists: The A field of the script is part of table B in the source database. The A field or the B table are no longer a part of the **gbase.dbb** destination structure.
- 'A' (Table 'B') was modified in 'C' (Table 'D') (formula 'E'):
The A field of the script is part of table B in the source database. The A field was associated in the **modifications.xml** file with the C field, which is part of table D in the **gbase.dbb** destination structure. The C field is populated using formula E.

Formula E was found in the **modifications.xml** file. A formula is displayed by the message when a Value attribute is different from the SQL name of a field.

Example (theoretic): 'dtEnd' ('amTicket' table) was modified in 'duration' ('amTicket' table) (formula 'dtEnd - dtStart')

 **Note:**

Double-clicking the mouse places the cursor on the problematic line.

 **Warning:**

There are no modification propositions made for the table names that are problematic.

Restore corrected application data

 **Note:**

Connect-It restores the corrected application data, which is transparent for the user if Connect-It is installed.

The restoration of application data can also be performed by AssetCenter Database Administrator or AssetCenter Script Analyzer.

Restoring application data corrected with AssetCenter Database Administrator

- 1 Launch AssetCenter Database Administrator 4.1.0.
- 2 Connect to the backup (1) of the working database with the **Admin** login (**File/ Open/ Open existing database** menu).
- 3 Select the **Migration/ Restore the application data** menu.
- 4 Populate the **Migration folder** field: folder containing reference files necessary for the conversion.

There is one folder per database version that can be converted (usually **C:\Program**

Files\Peregrine\AssetCenter\migration\fromxxx, where **xxx** is the number of the earlier version).

- 5 Populate the **Working folder** field: folder that contains the application data exported with AssetCenter Database Administrator (tree structure of **.xml** files containing the application data to be manually converted).

This is the file that you specified in the **Working folder** field, via the AssetCenter Database Administrator menu **Migration/ Export application data**.

- 6 Click **OK**.
- 7 Consult the messages that appear on screen.
- 8 Consult the **sdurest.log** log file, which is located in the folder defined by the **Working folder** field.

 **Tip:**

You can use the Log viewer program to consult the log file.

Restoring application data corrected with AssetCenter Script Analyzer

- 1 Launch AssetCenter Script Analyzer.
- 2 Populate the **Working folder** field: folder that contains the application data corrected with AssetCenter Script Analyzer (tree structure of **.xml** files containing the scripts).
- 3 Select the **Action/ Restore the application data** menu.
- 4 Connect to the backup (1) of the working database with the **Admin** login.
- 5 Populate the **Migration folder** field: folder containing reference files necessary for the conversion.

There is one folder per database version that can be converted (usually **C:\Program Files\Peregrine\AssetCenter\migration\fromxxx**, where **xxx** is the number of the earlier version).

- 6 Populate the **Working folder** field: folder that contains the application data exported with AssetCenter Database Administrator (tree structure of **.xml** files containing the scripts).
This is the file that you specified in the **Working folder** field, via the AssetCenter Database Administrator menu **Migration/ Export application data**.
 - 7 Click **OK**.
 - 8 Consult the messages that appear on screen.
 - 9 Consult the **sdurest.log** log file, which is located in the folder defined by the **Working folder** field.
-



Tip:

You can use the Log viewer program to consult the log file.

Causes of rejection

- If the application data stored in an **.xml** file that is declared non-restored, the data is rejected.
- Any mandatory field in the version 4.1.0 must have a `Mapping` element in the **modifications.xml** file, or belong to a table that has not been modified since the earlier version, or have an unchanged SQL name between two tables associated with a `Mapping` element of the **modifications.xml** file.

If this condition is not respected, the conversion will not start.



Tip:

The mandatory nature of a field is defined by the **Mandatory** parameter in AssetCenter Database Administrator (with the value **Yes** or **Script**).

Verify the integrity of the backup (1) of the working database

Verify the integrity of the backup (1) of the working database as indicated in this guide's chapter **Step-by-step migration - simulating**

the conversion of the working database, section **Verify the integrity of the working database**.

Instead of connecting to the working database, you must connect to the backup (1) of the working database.

Select the option **Analyze only** instead of the **Repair** option.

If problems are displayed by the program, the conversion might not have been correctly performed.

You must then verify the conversion parameters, especially the **migration.xml** conversion file.

13 Verify restored application data

Restored application data is the data that you have verified and perhaps modified with AssetCenter Script Analyzer.

This does not guarantee that this application data will work when it is used with AssetCenter.

Only the manual testing of all application data will guarantee its proper functioning:

- 1 Display one by one the restored **.xml** files.
- 2 Locate the record that contains the restored application data.
- 3 Test the located application data in the backup (1) of the working database.

Tip:

You must, in particular, verify that the reorganization of the database structure has no impact on the record containing the application data to manually convert. (Just correcting the script does not suffice. For example: A workflow scheme using the Assets table might need to be reconfigured to take into account the addition of the Portfolio items table.)

-
- 4 When you have tested the restored application data, select the option **Functionally valid** in the AssetCenter Script Analyzer.

This means that you can restore the **.xml** file in the working database.

Queries - note

If a query identified a record linked by the value of its primary key, and if the records of this table have been moved to a new table using another index during the conversion, the query will no longer select the correct link.

Perform one of the following corrective procedures:

- Modify the primary ID in the query.
- Take advantage of the conversion to point the query to the value of a more stable field. This avoids the same problem reappearing during another conversion later on in your company's future.

14 Adapt the integration with external tools

If you integrated external applications with the AssetCenter database, you will probably have to adapt the integration mode of these applications.

The potentially concerned applications are listed in this guide's chapter **Step-by-step migration - converting the working database**, section **Finalize the backup (2) of the working database**, sub-sections:

- **AssetCenter Web**
- **Get-It**
- **Get-Resources**
- **Connect-It scenarios**
- **Import scripts**
- **Export scripts**

You only implement the new integration mode in these applications after the final conversion of the working database.

However, you still need to make preparations for this implementation now.

This enables you to limit the time required for this operation.

5 Step-by-step migration - converting the working database

CHAPTER

At this stage, you have:

- A **gbase.dbb** file where the structural changes you made to the database to convert were propagated.
- A **migration.xml** conversion file that was tested on a backup (1) of the working database.
- Manually converted application data that was tested in a backup (1) of the working database.

This chapter explains step-by-step which operations to perform to convert the working database.

15 Verify the integrity of the working database

Verify the integrity of the working database as indicated in this guide's chapter **Step-by-step migration - simulating the conversion of the working database**, section **Verify the integrity of the working database**.

16 Block and save the backup (2) of the working database

Blocking the database consists of prohibiting its use so that no modifications can be performed during the conversion (they might be lost).

Perform the following tasks:

- 1 Disconnect all users from the database.
- 2 Shut down the:
 - AssetCenter Server
 - AssetCenter APIs
 - External programs that access the database.
- 3 Block access to the database.
- 4 Make a backup (2) of the database as described in this guide's chapter **Step-by-step migration - simulating the conversion of the working database**, section **Copy (1) the working database**.

You need to keep the duration of this blockage as brief as possible in order to avoid problems for users.

This is why you need to take your time during the simulations that precede the real conversion to work out any issues.

17 Convert the backup (2) of the blocked, working database

To convert the backup (2) of the working database, follow the instructions described in this guide's chapter **Step-by-step migration - simulating the conversion of the working database**, section **Convert the backup (1) of the working database/ Converting the backup (1) of the database**:

- Instead of connecting to the backup (1) of the working database, you must connect to the backup (2) of the working database that you created just after blocking the working database.
- You use the **migration.xml** conversion file that you finalized on the backup (1) of the working database.

The actual conversion of the database should be as brief as possible, because the working database is blocked during this time.

If, despite the success of the previous simulations, you run into unexpected difficulties, you should:

- 1 Stop the conversion of the backup (2) of the working database.
- 2 Put this blocked database back into production.
- 3 Redo simulations with a new backup (1) of the working database.
- 4 Perform the migration process again, starting up from where you blocked the database.

18 Restore the manually converted application data

To restore the application data that was manually converted in the backup (2) of the working database, follow the instructions described in this guide's chapter **Step-by-step migration - simulating the conversion of the working database**, section **Processing application data to be manually converted/ Restore corrected application data:**

- Instead of connecting to the backup (1) of the working database, you must connect to the other backup (2) of the working database.
- You use the **.xml** files in the working folder that you corrected using the backup (1) of the working database.

19 Verify the integrity of the backup (2) of the working database

Verify the integrity of the working database as indicated in this guide's chapter **Step-by-step migration - simulating the conversion of the**

working database, section **Verify the integrity of the working database**.

Instead of connecting to the working database, connect to the backup (2) of the working database.

20 Finalize the backup (2) of the working database

Important:

The database we are talking about in this section is the backup (2) of the working database, which is blocked.

You will need to make alterations to the converted database for several reasons:

- Certain data will not have been converted by the conversion program. You must:
 - Restore the previously exported and corrected application data to manually convert outside of the working database.
 - Manually test and touch up certain data in the converted database.
- Certain functions have been added or improved upon.

To fully take advantage of this, you must prepare for the use of these functions in the converted database.

This provides an opportunity to improve upon the efficiency and the services performed by AssetCenter.

Finalization concerning all versions of the source database

Verifying the success of the conversion

We recommend that you verify that the conversion has been correctly carried out.

You can, for example:

- Scan the converted database in search of any obvious anomalies.
- Compare the number of records from certain tables before and after the conversion.

If there are any differences, they either correspond to purposeful specifications of the **migration.xml** conversion file or they are anomalies.

Modifications to the stored procedure **up_GetCounterVal**

This section concerns users who modified the stored procedure **up_GetCounterVal** in the source database.

Before converting the database, you need to have:

- 1 Manually updated the counters in the **amCounter** table that were diverted to other tables.
- 2 Restored the stored procedure **up_GetCounterVal** to its original state.

You can adapt the stored procedure **up_GetCounterVal** again according to the directives in the following technical notes:

- Microsoft SQL Server: TN317171736
- Sybase Adaptive Server: TN941931
- Oracle Workgroup Server: TN12516652
- DB2 UDB: TN1029175140

Triggers, indexes, stored procedures and views

Before the conversion, you put the earlier, working database back to its original state for everything concerning the modifications to these items.

Now you can manually perform these modifications again if they are still necessary.

Help on fields

The help on fields (and links) are stored in the **Help on fields** table (amHelp).

During the database conversion, the contents of this table are not modified.

Saving the customizations performed on the earlier version of the help on fields

- 1 Export the help on fields as they are.
 - 1 Start AssetCenter 4.1.0.
 - 2 Connect to the converted working database (**File/ Connect to database** menu).
 - 3 Display the list of records from the **Help on fields** (**Administration/ List of screens** menu).
 - 4 Configure the list so the fields and links appear in the order shown below:
 - Table (TableName)
 - Field (FieldName)
 - Description
 - Example
 - Precautions
 - 5 Export the contents of the list (**Export the list** shortcut menu).
- 2 Export the standard help on fields from the earlier version.
 - 1 Create an empty **Sybase SQL Anywhere** database.
To learn how to create an empty database, refer to the **Administering the database** guide, chapter **Creating an AssetCenter database**, section **Creating an empty AssetCenter database using your DBMS**, sub-sections **Creating a database at the DBMS level** and **Runtime Sybase SQL Anywhere** .
 - 2 Start the earlier version of AssetCenter.
 - 3 Connect to the empty database (**File/ Connect to database** menu).
 - 4 Display the list of records from the **Help on fields** (**Administration/ List of screens** menu).
 - 5 Configure the list so the fields and links appear in the order shown below:

- Table (TableName)
- Field (FieldName)
- Description
- Example
- Precautions

6 Export the contents of the list (**Export the list** shortcut menu).

3 Compare the two exported files.

The differences correspond to the modifications that you made.

Conserve a copy of these modifications.

Update the help on fields in the version 4.1.0.

- 1 Start AssetCenter Database Administrator.
- 2 Select the **File/ Open** menu.
- 3 Select the **Open database description file - create new database** option.
- 4 Select the **gbbase.dbb** file, located in the **config** sub-folder of the AssetCenter 4.1.0 installation folder.
- 5 Select the **Action/ Create database** menu.
- 6 Select the converted database (**Database** field).
- 7 Clear the option **Create database**.
- 8 Clear the option **Create system data**.
- 9 Clear the option **Use AutoCAD integration**.
- 10 Select the **Use help on fields** option.
- 11 Clear the option **Import extra data**.
- 12 Click **Create**.

Reapplying customizations to help on fields

By updating the help on fields of version 4.1.0, you are overwriting the customizations that you already did.

You can thus redo this customization manually using the copy you saved of these customizations of the earlier version's help on fields.

You can, for example, import your modifications using the **Table** and **Field** fields (TableName and FieldName) as reconciliation keys.

Fields populated arbitrarily

There are other fields that are populated arbitrarily during the conversion. This is due to a lack of relevant information.

The way these fields are populated is defined in the **migration.xml** conversion file.

In order to easily find these fields after the conversion, they are populated by concatenating the ^ character with other values taken from the database.

You can verify the value of these fields for all records concerned and modify them if necessary.

Given the number of important records potentially concerned, such a modification can be performed by an export, followed by an import, of records to modify.

The can concern, depending on the tables, the following fields:

- **Code** (Code)
- **Bar code** (BarCode)
- **SQL name** (SQLName)
- **Full name** (FullName)
- Etc.

To obtain an exhaustive list of fields to verify:

- 1 Open the **migration.xml** file used for the conversion in the text editor.
- 2 Search for the ^ character.

You will obtain a list of the fields to verify.

For example:

```
<Mapping to="amAssetRent" from="amAssetRent">
  <Field sqlname="Code" value="'^' || SDUSTR lAssetRentId"/>
</Mapping>
```

In this example, you must verify the value of the **Code** field in all the records of the **amAssetRent** table whenever the value starts with ^

AssetCenter Web

You must upgrade AssetCenter Web in version 4.1.0.

If you only use the standard pages of AssetCenter Web, this operation will suffice: You can use the new standard pages of AssetCenter Web.

If you created additional Web pages or customized standard Web pages:

- 1 Save the previous additional or customized pages.
- 2 Upgrade AssetCenter Web to the version 4.1.0.
- 3 Test and adapt each Web page one after the other.

Output events

Records from the **amOutputEvent** table are not modified during the conversion.

Their values may reflect the structure of the database before conversion.

You must therefore finish the conversion manually.

 **Note:**

The records in the **amInputEvent** table are not modified during the conversion. Unlike the **amOutputEvent** table, this will never pose problems.

Get-It

For each Web application developed with Get-It to function with the AssetCenter 4.1.0 database:

- 1 Verify that your version of Get-It is listed in the AssetCenter 4.1.0 compatibility matrix (available on the Peregrine Systems customer support Web site).
- 2 Upgrade Get-It if necessary.
- 3 Test and adapt each customized Web page one after the other.

Get-Resources

For Get-Resources to function with the AssetCenter 4.1.0 database:

- 1 Verify that your version of Get-Resources is listed in the AssetCenter 4.1.0 compatibility matrix (available on the Peregrine Systems customer support Web site).
- 2 Upgrade Get-Resources if necessary.

If you only use the standard pages of Get-Resources, this operation will suffice: You can use the new standard pages of Get-Resources.

If you created additional Web pages or customized standard Web pages:

- 1 Save the previous additional or customized pages.
- 2 Upgrade Get-Resources if necessary.
- 3 Test and adapt each customized Web page one after the other.

Connect-It scenarios

To access the AssetCenter 4.1.0 database using Connect-It, you must use the version of Connect-It provided with AssetCenter 4.1.0.

If you use standard Connect-It scenarios, you must now use the new standard scenarios.

If you created your own scenarios:

- 1 Save the previous non-standard scenarios.
- 2 Upgrade Connect-It.
- 3 Open each scenario one by one in Connect-It.
- 4 For each scenario:
 - 1 Examine the possible warning messages displayed by Connect-It when you open a scenario.
 - 2 Correct the scenario according to the warning messages.
 - 3 Execute the scenario using test data.
 - 4 Correct the possible problems that present themselves during this test.

Import scripts

You must test, one after the other, each import script that you have created and want to keep:

- 1 Launch AssetCenter 4.1.0.
- 2 Connect to a test database (which can be a backup of your working database).
- 3 Launch the import module (**File/ Import** menu).
- 4 Select the **Import database** option.
- 5 Select the **Text** tab and click **Open**.
- 6 Open the script in the new window that appears (**File/ Open script** menu).
- 7 Verify each association one at a time (double-click on couples (**source, destination**) in the right-hand list).
- 8 Save your modifications (**File/ Save**).
- 9 Test the import (**Import** button).
- 10 Correct the import script again if necessary.

Export scripts

You must test each export script that you have created and want to keep:

- 1 Launch AssetCenter Export 4.1.0.
- 2 Connect to the working database (the export does not modify the data in the database to which you are connecting).
- 3 Open the script (**File/ Open script** menu).
- 4 Verify each query one at a time.
 - 1 Select the query in the upper-hand list.
 - 2 Click the **Magnifying glass** icon in the bottom-hand list.
 - 3 If the query is valid, no warning message will appear.
 - 4 If the query is not valid, a warning message will appear.
 - 5 Whether a warning message appears or not, you must verify that the query parameters still correspond to what you were expecting (taking into account that the database structure has changed).

For example: Data that you were searching in the Assets table might now be now located in the Portfolio items table.

- 5 Save the modifications (**File/ Save script**).
- 6 Test the export (**Actions/ Execute script**).
- 7 Correct the export script again if necessary.

Customizing the database and the detail of screens

In version 4.1.0, AssetCenter offers the possibility to add tables, fields, links, buttons and tabs.

You can take advantage of the migration to add such application data.

To learn how to customize fields, refer to the **Administering the database** guide, chapter **Customizing the database**.

Forms

When you convert the working database, the forms are left as they are. The changes to the database structure are not recovered.

It is probable that several SQL names of tables, fields and links are no longer valid.

Test each form one at a time:

- 1 Launch AssetCenter 4.1.0.
- 2 Display the list of forms (**Tools/ Reporting/Forms**).
- 3 Select each form one at a time.
 - 1 If a warning appears, read it and correct the form according to its message.
 - 2 Print the form and examine its results.
 - 3 Modify the form if necessary.

Views

When you convert the working database, the views are left as they are. The changes to the database structure are thus not recovered.

Since views memorize applied filters and columns to be displayed, you need to verify the views by displaying them one after the other. For each view, validate the selection of columns to be displayed as well as any filters applied:

- 1 Launch AssetCenter.
- 2 Select each view one at a time (**Tools/ Views** menu).
- 3 If a warning appears, read it and correct the view according to its message.



Tip:

Create any new view that you will need.

Crystal Reports

During the database conversion, the reports are left as they are. The changes to the database structure are thus not recovered. It is probable that several SQL names of tables, fields and links are no longer valid.

Reusing previous reports

- 1 Launch AssetCenter.
- 2 Display the list of reports (**Tools/ Reporting/ Reports** menu).
- 3 Delete the reports that you no longer want to keep.
- 4 Test each report that you want to keep one at a time.
For each report:
 - 1 Place your cursor in the context of that report (the list or details of an asset, for example).
 - 2 Display the screen for printing reports (**File/ Print**).
 - 3 Populate the **Type** field according to the type of report you want to test.
 - 4 Select the report.
 - 5 Click **Preview**.

- 6 If a warning appears, read it and correct the report in Crystal Reports according to its message.
 - 5 If you want to import the new, standard reports provided with AssetCenter 4.1.0:
Modify the SQL name of the previous reports that you will keep before importing the new reports.
-

 **Warning:**

If you do not do this, the previous reports will be overwritten by the new reports with the same SQL name.

Deciding not to use previous reports

- 1 Launch AssetCenter.
- 2 Display the list of reports (**Tools/ Reporting/ Reports** menu).
- 3 Delete all the previous reports.

Importing the standard reports provided with AssetCenter 4.1.0

To import the **sample data** reports in the converted database:

- 1 Start AssetCenter Database Administrator.
- 2 Select the **File/ Open** menu.
- 3 Select the **Open database description file - create new database** option.
- 4 Select the **gbbase.dbb** file, located in the **config** sub-folder of the AssetCenter 4.1.0 installation folder.
- 5 Select the **Action/ Create database** menu.
- 6 Select the converted database (**Database** field).
- 7 Clear the option **Create database**.
- 8 Clear the option **Create system data**.
- 9 Clear the option **Use help on fields**.
- 10 Clear the option **Use AutoCAD integration**.
- 11 Select the option **Import extra data**.

- 12 Populate the **Data to import** list by selecting **Rapports Crystal Reports**.
- 13 Click **Create**.

User rights, access restrictions and user profiles

Since new tables, fields and links have been added to the database structure, you must adapt your user rights, access restrictions and user profiles.

You must verify the queries corresponding to each access restriction.

You just need to add the new tables to the existing user rights and profiles and to create new rights and restrictions if necessary.

Line-of-business data

To avoid having to create certain reference data yourself, AssetCenter 4.1.0 is installed with **line-of-business data** that you can import or insert in your database if you find them to be useful:

- 1 Start AssetCenter Database Administrator.
- 2 Select the **File/ Open** menu.
- 3 Select the **Open database description file - create new database** option.
- 4 Select the **gbbase.dbb 4.1.0** file in which you have propagated the structural changes.
- 5 Select the **Action/ Create database** menu.
- 6 Select the converted database (**Database** field).
- 7 Clear the option **Create database**.
- 8 Clear the option **Create system data**.
- 9 Clear the option **Use help on fields**.
- 10 Clear the option **Use AutoCAD integration**.
- 11 Select the option **Import extra data**.
- 12 Populate the **Data to import** list by selecting the **line-of-business data** that interests you.
- 13 Click **Create**.

Finalization concerning only those source database versions later than version 4.0.0

Units

During the conversion, the **Dimension** and **Symbol** fields of the **Units** table (**amUnit**) were populated using different sources.

You can verify the values that were created here and correct them if necessary.

Brands created from product families

During the conversion, the **amFamily** table is transferred to the **amBrand** table.

Verify the values of the **Name** and **FullName** fields of the **amBrand** table for the records that originate from this conversion.

To identify these records, search those records whose **Name** field contains the character ^.

Countries

During the conversion, the **FullName** and **Name** fields of the **amBrand** table were populated using different sources.

You can verify the values that were created and correct them if necessary.

Brands, units and countries

Since the version 4.0.0, brands, units and countries are populated by the link to the **amBrand**, **amUnit** and **amCountry** tables. They are no longer populated by a field linked to an itemized list.

During the conversion of fields and links, records are created in the **amBrand**, **amUnit** and **amCountry** tables.

It might occur that certain records created this way will be nearly identical.

You might find that certain values do not correspond to the norms that you established for the most recent itemized lists. In effect, you can

delete a value from an itemized list without effecting the records in the database already set to this about-to-be-deleted value.

Example: **H.P.** and **Hewlett Packard**.

You can take advantage of this conversion to get rid of any double records by sorting the records according to the **Name** field.

Natures

Name and Code fields

During the conversion, the **Name** and **Code** fields of the **Natures** table (amNature) were populated using different sources.

You can verify the values that were created here and correct them if necessary.

Natures created from software installations

All the sub-natures of the **Software** nature must be reorganized according to your intended organization.

 **Note:**

The **Software** nature is used to reattach the software installation models.

Models created from software items

During the conversion, the **amSoftware** table is transferred to the **Models** table (amModel).

The models created in this manner are attached to a root model whose **Name** field is set to **^amSoftware**.

Verify the models attached to the **^amSoftware** model.

You can rename the **^amSoftware** model.

Assets created from license contracts

This section concerns users who created license contracts.

During the conversion, license contracts are transformed into assets that are linked to a model named **^amSoftLic**.

This model is, itself, linked to a nature named **^amSoftLic**.

You can:

- 1 Search all the assets linked to the model named **^amSoftLic**.
- 2 Check if there is a model more relevant to which you can link these assets.
- 3 For the assets not having a more relevant model, rename the model and the nature.

Locations

During the conversion, the addresses of the **Suppliers** table (**amCompany**) were moved to the **Locations** table (**amLocation**).

The locations created in this manner are attached to a root location whose **Name** field is set to **^amCompany**.

Verify the locations attached to the **^amCompany** location.

Rename the **^amCompany** location if you consider this to be useful.

Budgets

If you activated the **Mapping** elements in the **migration.xml** files that associate the **amBudget** table to the **amBudgLine** table, records will be created somewhat haphazardly in the following tables:

- amBudget
- amPeriod
- amFYDivision
- amFinancialYear
- amBudgClass
- amBudgCenter
- amBudgLine
- amBudgetCategory

Clean up all of these tables.

Verify the budget whose **Name** field is set to **^amBudget**.

Verify the budget classification whose **Name** field is set to **^amBudgClass**.

Reorganize the periods thus created into coherent time divisions.

 **Note:**

During the conversion, no time divisions are created.

In particular, make sure that the periods belonging to a time division cover the financial year in full without overlapping each other.

Cost categories created from budgets

During the conversion, the **amBudget** table is transferred to the **amCostCategory** table.

Budgets having the same name during the conversion change names. This is so the obtained cost categories all have different names.

Verify the **Name** field and change it if necessary.

To find this information again, search the records whose **Name** field contains the ^ character.

Functional domains

During the conversion, the **SQL name** field (SQLName) is populated by simply recopying the value of the **Name** field.

The obtained SQL name does not necessarily conform to the norms established for this type of field. (Only letters of the standard alphabet, numbers and the "_" character are authorized.)

You must verify each SQL name and, if necessary, modify it so it conforms to the norms.

Functional rights

During the conversion, the following fields of the **amEmplDept** table were deleted from the database structure:

- bEstimRight
- bHDCloseTickRight

- bHdProceedRight
- bHdSaveCallRight
- bOrderRight

The value of these fields was not migrated to any fields of the 4.1.0 database.

You can:

- 1 Identify the employees in the unconverted database whose fields were populated.
- 2 Create functional rights that fulfill the same function as these deleted fields.
- 3 Attach employees to the appropriate functional rights.

Catalog references

Verify the records in the **Catalogs** table (**amCatalog**).

In particular, verify the record in the **Catalogs** table (**amCatalog**) whose **Name** field (**Name**) is set to **OffCatalog**.

This record contains the references (**amCatRef** table) created from the converted records of the **amPordLine** table.

Features that were linked to license contracts

During the conversion, certain license contracts (**amContract**) were transformed into portfolio items (**amPortfolio**).

This process is described in this guide, chapter **Step-by-step migration - simulating the conversion of the working database**, section **Convert the backup (1) of the working database/ License contracts**.

It is possible that the features used to describe the license contracts are no longer used in the **amContract** table.

Verify this by searching the features (**amFeature**) linked to feature parameters (**amFeatParam**) concerning the **amContract** table.

Delete the features and feature parameters that are no longer used.

Purchase orders

Due to a lack of sufficiently accurate information in the source database during the conversion, the **seStatus** field of the records in the **amPOrder** table are set to **Quoted** if the order was created from an estimate. In all other cases, it is set to **Ordered**.

You can verify the status of all orders created in the **amPOrder** table.

Reorganizing the repository

The database model that structures the AssetCenter repository has greatly changed.

To use data in good condition and take advantage of the new possibilities offered by AssetCenter, you must:

- 1 Understand the new data model.
For this, refer to the **Portfolio** guide, chapter **Overview (Portfolio)**.
- 2 Verify and if necessary refine the contents of the following tables:
 - Natures (amNature)
 - Models (amModel)
 - Brands (amBrand)
 - Assets (amAsset)
 - Portfolio items (amPortfolio)
 - Products (amCatProduct)
 - Catalog references (amCatRef)
 - Catalogs (amCatalog)
 - Requests (amRequest)
 - Computers (amComputer)
 - Telephones (amPhone)
 - Software installations (amSoftInstall)
- 3 Understand the impacts these structural changes have on your use of the Procurement module.

 **Note:**

AssetCenter 4.1.0 uses a new concept of overflow tables to move certain data to peripheral tables. For example, information about portfolio items coming from inventory scanning tools are stored in an overflow table. The appearance of these overflow tables means the certain fields have also been moved to these tables.

- Computers (amComputer)
 - Telephones (amPhone)
 - Software installations (amSoftInstall)
-

Chargeback and budget tracking

From version 4.0.0 onwards, AssetCenter increases the possibilities of processing cost accounting and budgetary data.

To use this data in good condition and take advantage of the new possibilities offered by AssetCenter, you must:

- 1 Understand how the Financials module works.
For this, refer to the **Financials** guide, chapter **Expenses**, section **Introduction to expenses**.
- 2 Verify and refine the contents of the tables linked specifically to the Financials module.

To obtain a list of these tables, refer to the **Financials** guide, chapter **References**, section **Tables (Financials)**.

6 Step-by-step migration - final phase

CHAPTER

This chapter explains step-by-step which operations to perform to get your working database up and running after the conversion.

Updating AssetCenter programs

You must upgrade all the AssetCenter programs on all administration and user machines.

You must also make sure that the version of the programs that interact with AssetCenter are still compatible with AssetCenter 4.1.0. If necessary, upgrade these programs as well.

To obtain a list of AssetCenter programs and programs that interact with AssetCenter, refer to the **Installation** guide, chapter **List of AssetCenter programs**.

To learn which program versions are compatible with AssetCenter 4.1.0, consult the Peregrine Systems customer support site.



Tip:

Certain compatibility information exists as well in the **Installation** guide's chapter **After having installed the AssetCenter programs**.

Install AssetCenter Server on an administration machine

AssetCenter Server carries out a number of automatic tasks on the AssetCenter database. If it is not launched, AssetCenter cannot function correctly.

You must therefore:

- 1 Install AssetCenter Server on a client machine.
- 2 Properly configure AssetCenter Server.
- 3 Execute AssetCenter Server permanently.

To learn more about how AssetCenter Server works, refer to the **Administering the database** guide, chapter **AssetCenter Server**.

Delete the AssetCenter caches of your previous database

If you use a cache with the connection to your previous database, we recommend that you delete this cache.

To learn more about how caches work, refer to the **Using AssetCenter** guide, chapter **Reference information**, section **Connections**, sub-section **AssetCenter performances**.

Upgrade AssetCenter programs

To upgrade the programs, install AssetCenter 4.1.0.

The installation program will detect the existence of an earlier version of AssetCenter. If it detects one of them, it proposes that you either replace the earlier version or install AssetCenter in a new folder.

 **Note:**

"Replacing" consists of removing the previous version, then installing the new version as if the previous version did not exist (the only limitation: AssetCenter 4.1.0 is installed in the same folder as the previous version).

If the installation program tries to install a Sybase SQL Anywhere database (file extension **.db**) and there is already a file with the same name on the machine, you will be asked for confirmation before the file is overwritten.

 **Warning:**

Do not confirm the replacement if you want to keep the previous database!

To learn about the installation procedure (precautions to take, steps to follow, ways to install AssetCenter), refer to the **Installation** section of this guide.

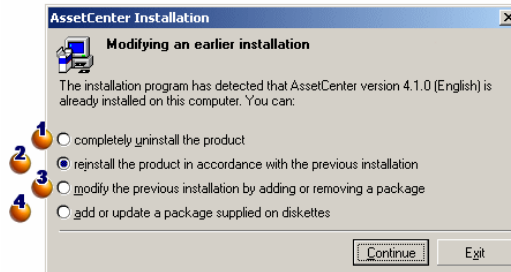
 **Tip:**

If you are installing AssetCenter 4.1.0 on a conversion machine, be sure to conserve your previous version of AssetCenter for the time being.

Installation program screens

The following is selection of screens from the installation program which may raise questions during the upgrade.

This screen appears when a previous version 4.1 of AssetCenter is already installed:

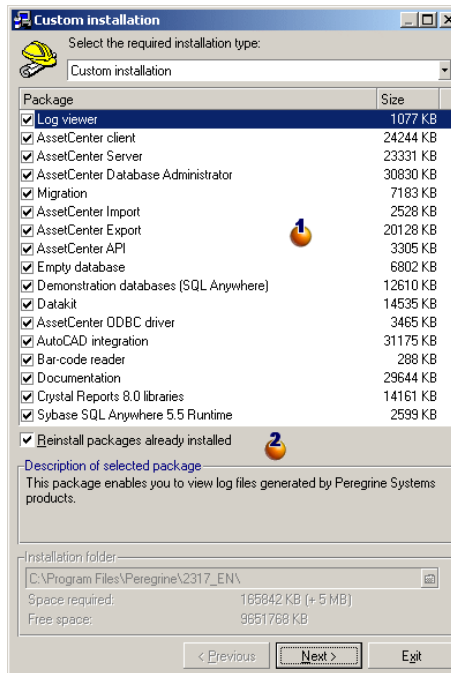


1 and **4**: These options are not to be used during the upgrade.

2: Select this option if you are certain not to have any packages to add or remove according to the previous installation.

3: Select this option if you want to add or remove packages according to the previous installation, or if you are not sure. This option is particularly useful if you do not want to install certain packages such as AssetCenter Server on a client machine.

The following screen appears if you selected the option **Modify the previous installation ...** in the previous screen.



1: The packages already installed are selected. Select those that you want to add. Leave selected those you want to reinstall. Clear those that you do not want to install.

2: If you select this option, the selected packages and already installed packages will be reinstalled. If you clear this option, only the new packages that are selected will be installed.

Note:

To automate these procedures, you can perform an unattended installation from the command line. To learn more about this, refer to the **Installation** guide, chapter **Installing the AssetCenter programs**, section **Automatic installation (command line)**.

Verify that AssetCenter can be launched without problems

If you are having problems launching AssetCenter 4.1.0, contact user support.

Remove the old connections to databases and create new ones

The objective is to have the users connect to the backup (2) of the converted database.

Refer to the **Using AssetCenter 4.1.0** guide, chapter **Reference information**, section **Connections**.

If you prefer, you can modify the previous connections.

Create an AssetCenter cache for your connections if you consider this will be useful.

Modify the customizations of AssetCenter at the level of the client machines if you consider this to be useful

To do this, refer to the **Using AssetCenter 4.1.0** guide, chapter **Customizing a client machine**.

Putting the copy (2) of the converted database into production

This is the last step of the migration process.

You have already:

- Totally converted and touched up the backup (2) of the working database.
- Upgraded the AssetCenter programs on all user and administration machines.

Now you must perform the following tasks:

- 1 Put AssetCenter Server into production on the converted and finalized backup (2) of the working database.
- 2 Relaunch the external programs that access the working database.
- 3 Inform users that they can use the database.

7 Glossary (Migration)

CHAPTER

Migration

The migration is a set of operations required to convert an earlier version of AssetCenter of the version 4.1.0:

Migration includes:

- Converting the working database (structure and contents) in order to make it compatible with the 4.1.0 version of AssetCenter.
- Updating the AssetCenter programs to the 4.1.0 version on all administration and user machines.

Updating AssetCenter programs

One of the operations required by the AssetCenter migration.

Updating the programs involves reinstalling all the AssetCenter programs on all administration and user machines so that they are a version 4.1.0.

Do not confuse with ...

Converting the working database

Converting the working database

One of the operations required by the AssetCenter migration. Converting the working database involves modifying its structure and contents in order to make it compatible with the 4.1.0 version of AssetCenter.

The conversion is performed in several steps. Certain steps are performed manually, others with the use of additional tools.

Do not confuse with ...

Updating AssetCenter programs

Conversion file

A conversion file is a file that describes which data to transform during the database conversion and what transformations to perform.

The conversion files are named **migration.xml**.

They are generally located in the **C:\Program Files\Peregrine\AssetCenter\migration\fromxxx** folder.

AssetCenter is installed with conversion files by default (1 file per version of AssetCenter that is supported by the migration).

You can customize these files.

Conversion machine

The conversion machine is the computer you use to convert the working database to the 4.1.0 format.

This computer requires a specific configuration, which is described in this guide.

Working database

The working database is the AssetCenter database that you use to manage your portfolio.

Do not confuse with ...

Demonstration database

Trigger

A trigger is an action that is automatically "triggered" by AssetCenter when a database field or link is modified.

8 References (Migration)

CHAPTER

Adapting the migration.xml conversion file

Warning

 **Warning:**

Adapting the conversion file requires strong technical skills, an in-depth understanding of the source version of AssetCenter, as well as the 4.1.0 version. Thus, the adaptation of the conversion file can only be done by a Peregrine Systems-certified engineer.

All modifications of the conversion file made by an uncertified person are done under the sole responsibility of the person making the modification(s), and not under the responsibility of Peregrine Systems.

 **Tip:**

Keep in mind that Peregrine Systems and its partners can provide specialized and experienced consultants who can adapt this conversion file for you.

This reference section is intended for certified engineers only.

 **Important:**

When you customize the **migration.xml** conversion file, you must neither rename it nor replace it. This is because the tools that use this file will search for it in the standard folder.

We also recommend that you make a backup of this conversion file before starting to modify it.

Reminders

To learn more about conversion files, refer to this guide's chapter **Glossary (Migration)**, section **Conversion file**.

To learn when a conversion file needs to be adapted, refer to this guide's chapter **Step-by-step migration - simulating the conversion of the working database**, section **Adapting the migration.xml conversion file**.

What does the conversion file do?

The conversion file defines the rules for converting fields whose values cannot be conserved as they are because:

- The table to which the field belongs has disappeared or changed its SQL name.
 - The field has disappeared or changed its SQL name.
 - The field is part of a feature that you want to transfer to a direct field or a table in the 4.1.0 database.
-

 **Note:**

The links are processed via foreign keys (which are actually fields).

The conversion file is used to generate SQL commands for modifying the working database (SQL used for the DBMS).

Conversion rules

Certain conversion rules are automatically determined by the conversion program:

- If a table's structure is identical between the earlier version and the 4.1.0 version of AssetCenter (the SQL names, fields, links and indexes are the same):

The fields do not need to be declared in the conversion file: Their values will not change.

 **Tip:**

You can, however, define conversions for the fields and links of a table that is structurally unchanged if you need to.

- If the SQL names of the fields are the same for the associated source and target tables in a Mapping element of the conversion file:

These fields are automatically associated. You do not need to cite them in the conversion file unless you want to modify their values.

Syntax of the conversion file

Global syntax

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE MigrationFile SYSTEM "acmig.dtd">
<MigrationFile continueonerror=[AA]>
  <StartScript engine='[G] '>
    [A]
  </StartScript>
  <Translate table="[R]" into "[S]"/>
  <Mapping to="[C]" from="[B]" where="[K]" orderby="[O]"
groupby="[P]" having="[Q]" autofill="[L]">
  <PreActions engine='[T] '>
    [U]
  </PreActions>
  <Field sqlname="[E]" value="[F]" translate="[X]"
feature="[Y]" featuretable="[Z]">
    <Exception engine='[M]' value="[N]"/>
```

```

</Field>
<PostActions engine=' [V] '>
  [W]
</PostActions>
</Mapping>
<Script engine=' [O] '>
  [I]
</Script>
<!-- [J] -->
<!-- [P] ---->
</MigrationFile>

```

;<?xml version="1.0" encoding="iso-8859-1"?> **line**

This line is mandatory.

It cites the XML version as well as the character set used in the file.

You can modify this character set, but only if it corresponds to the character set used in the **.xml** file.

<!DOCTYPE MigrationFile SYSTEM "acmig.dtd"> **line**

This line indicates which **.dtd** to associate to the **.xml** file.

AssetCenter installs the **acmig.dtd** file next to the **migration.xml** conversion files.

The **acmig.dtd** is not mandatory, but it is useful to validate the structure and make it easier to read the **.xml** file.

The **acmig.dtd** requires the use of an XML editor in order to be active.

MigrationFile element

This element contains three elements that describe the operations to perform during the conversion:

- StartScript
- Translate
- Mapping

- Script

continueonerror attribute

This attribute is optional.

When AA is set to No, the conversion is interrupted at the first sign of a conversion error.

When AA is set to Yes, the conversion continues as long as possible despite any errors found during the conversion.

By default, this attribute is set to No.

engine attribute

This optional attribute is used by several elements to define the DBMS to which the element is applied.

Possible values:

- Sybase
- MSSQL
- Oracle
- DB/2
- SQLAnywhere

You must respect the case.

SQLAnywhere applies to the full version, not the runtime version (which does not support the SQL commands necessary for the conversion).

StartScript element

This element contains an [A] SQL script, which you will execute before the database conversion (and even before you rename the previous tables).

The advanced users will execute such a script to remove the customizations made to the structure of the previous database and to deactivate its triggers, etc.

The script must be written in an SQL language conforming to the one used in the DBMS of the AssetCenter database.

 **Tip:**

There is one exception to this constraint: To concatenate strings, you can use the || operator with all engines (it is transformed into + for MSSQL and Sybase).

 **Warning:**

The AQL language of AssetCenter is not recognized.

Each SQL command line is executed using a GO line.

For example:

```
UPDATE amPortfolio SET lParentId=0 WHERE lPortfolioItemId
IN (SELECT p.lPortfolioItemId FROM amAssetOld a, amPortfolio
p WHERE a.lParentId=0 AND p.lAstId=a.lAstId)
GO
DELETE FROM amItemListVal WHERE lItemId=(SELECT
lItemId FROM amItemizedList WHERE Identifier='amBrand')
GO
```

engine attribute

The StartScript element with the engine attribute replaces the StartScript element without the engine attribute when the StartScript element is executed on a database where the DBMS is [G].

Translate element

This element is used during the conversion of fields that store table names (an action's context, for example).

A Translate element must be defined when a source table [R] is associated with several destination tables [S] inside several Mapping elements.

The `Translate` element is used to indicate which of these `[S]` tables is the destination table for the automatic conversion of fields that store table names.

The conversion of fields that store table names uses a mapping table, which is automatically created at the onset of the conversion using information in the **migration.xml** conversion file.

The mapping table maps:

- The tables associated in a `Mapping` element by the `to="[C]"` and `from="[B]"` attributes when tables `[C]` and `[B]` are different.
- The tables associated in a `Translate` element by the `table="[R]"` and `into "[S]"` attributes.

The associations performed from `Translate` elements take superiority over those performed from `Mapping` elements.

The `maptable` is used by a conversion-file script using the `UPDATE` command.

This enables the replacement of the old table name by the new table name:

Example:

```
UPDATE amDocument SET DocObjTable = ( SELECT newsqlname FROM
sdutrans WHERE oldsqlname = amDocument.DocObjTable ) WHERE
amDocument.DocObjTable IN( SELECT oldsqlname FROM sdutrans)
```

Mapping element

This element enables you to transfer and convert the fields of a table in the previous structure to a table in the version 4.1.0 structure.

from attribute

The `from` attribute is mandatory. It identifies the `[B]` table of the previous structure.

In the case of a join, several tables can be used by respecting the following syntax:

```
from="[SQL name of table 1] alias1, [SQL name of table 2]
alias2, ..., [SQL name of table n] aliasn"
```

to attribute

The `to` structure is mandatory. It identifies the [C] table of the new structure.

where attribute

The `where` attribute is optional. It specifies the [K] SQL condition, which defines the records of the [B] table that must be processed by the Mapping element.

By default, the `where` clause excludes the null primary-key record from source table [B] (internal join - where [SQL name of the primary key] <> 0).

By default, the `where` clause includes null primary-key records from remote tables linked to table [B] (external join).

For example, in the following association:

```
<Mapping to="amCatProduct" from="amProdSoftInfo s, amSoftware
soft" where="s.lSoftId = soft.lSoftId">
```

The records for which `s.lSoftId` and `soft.lSoftId` are equal are retained.

To learn about what null primary-key records do, refer to the **Advanced use guide**, chapter **AQL queries**, section **Recommendations for writing AQL queries**, sub-section **Reason for and usefulness of primary key 0 records**.

orderby attribute

The `orderby` attribute is optional. It specifies the order of the SQL sort [O].

groupby attribute

The `groupby` attribute is optional. It specifies the [P] SQL sub-set.

having attribute

The `having` attribute is optional. It specifies the [Q] SQL search conditions.

autofill attribute

The `autofill` attribute is optional. It can accept either `yes` or `no` as its value. By default, its value is `yes`.

When its value is `no`, only the fields of the [C] table processed by a `Field` element are populated.

The fields automatically associated by the conversion program are not populated. (These are the fields whose SQL name is the same in tables [B] and [C].)

PreActions element

This element contains an SQL script [U] to execute before executing the `Field` element that follows it.

This element's syntax is the same as for a `StartScript` element.

The advanced users will execute such a script in order to perform operations that cannot be done using a `Mapping` element.

At the time you execute the `PreActions` element, the previous tables are not yet deleted.

You can thus still use the previous data.

The `PreActions` element is intended for users who have modified the standard structure of the database to be converted.

Field element

This element enables you to populate the new SQL name field [E] with the value calculated by the SQL expression [F].

The SQL expression [F] must rely on fields from the [B] table identified by their SQL name.

If the SQL expression [F] is not valid for a given DBMS, you must populate the `Exception` element just after the `Field` element line.

feature attribute

This attribute is used to convert a source feature value to a field in the destination database.

This attribute's [Y] value corresponds to the SQL name of the feature whose values are to be converted.

featuretable attribute

This attribute is used to convert a source feature value to a field in the destination database.

This attribute's [Z] value corresponds to the SQL name of the table that stores the feature values to be converted.

Warning:

The table that stores the feature values that are associated to it in the [Z] table must be declared at the level of the `from` attribute in the `Mapping` element.

For example: The `amFVAsset` table stores the features values that are associated to its records in the `amAsset` table. If you want to convert the [Y] feature values in a field, you must declare the `amAsset` table at the level of the `from` attribute. And you must declare the `amFVAsset` table at the level of the `featuretable` attribute.

Exception element

This element enables you to create an exception specific to a given DBMS for the `Field` element that precedes it.

engine attribute

The `engine` attribute enables you to define the [O] DBMS to which the exception applies.

The `Exception` element replaces the `Field` element for the [O] DBMS.

value attribute

The `value` attribute enables you to define the SQL expression that is valid for the [O] DBMS.

In the case of a join, the alias must be used according to the following syntax:

```
value="[alias of the table].[SQL name of the field]"
```

SDU_NEWID variable

This variable is sometimes used by the `value` attributes that define new values for the primary keys.

`SDU_NEWID` is the value of the primary key ID having the largest numeric value in the database before the conversion and increased by 1.

`SDU_NEWID` is automatically calculated by the conversion program.

PostActions element

This element contains an SQL script [W] to execute after executing the `Field` element that precedes it.

This element's syntax is the same as for a `StartScript` element.

The advanced users will execute such a script in order to perform operations that cannot be done using a `Mapping` element.

At the time you execute the `PostActions` element, the previous tables are not yet deleted.

You can thus still use the previous data.

The `PostActions` element is intended for users who have modified the standard structure of the database to be converted.

Script element

This element contains an [I] SQL script to execute after having executed the Mapping elements, but before deleting the previous tables that are now obsolete.

This element's syntax is the same as for a StartScript element.

The advanced users will execute such a script in order to perform operations that cannot be done using a Mapping element.

At the time you execute the Script element, the previous tables are not yet deleted.

You can thus still use the previous data.

The Script element is intended for users who have modified the standard structure of the database to be converted.

!-- element

This tag enables you to insert a [J] comment in the code. This comment will not be taken into account by the conversion program.

!-- element

This tag enables you to insert a [J] comment for the user of the conversion file. This comment will not be taken into account by the conversion program.

Using special characters

Here are the indications for using certain characters that can be interpreted in a particular manner.

These indications are not exhaustive. For more information, we recommend that you consult SQL and XML documentations.

In general, the general structure of the conversion file must respect XML constraints, and the attribute values must respect SQL constraints.

Here are some characters whose interpretation is particular:

Special character	Interpretation	Example	Equivalent when the character must be interpreted as text.	Example
"	Delimits the value of an XML attribute.	value="lAssetRentId"	\ "	value="\ \""
'	Delimits the SQL text string inside the value of an attribute.	value="soft.Publisher+'/' +soft.Name"	' '	value="''''''"
<	Opens an XML tag.	</Mapping>	<	value="<";"
>	Closes an XML tag.	</Mapping>	>	value=">";"
&	Marks the beginning of an entity.	<	&	value="&";"
;	Marks the end of an entity.	<	; without & before	value="";"
\	SQL escape character.		\\	value="\\ \""
	SQL string concatenation character (valid for all DBMSs).	value="'A' 'B' "	' ' ' '	value="'A' ' B' "

Dividing the fields of an previous table between several new tables

For example, the earlier version of AssetCenter used the Assets table. In this version, there is a Portfolio items table and an Assets table. Thus, the fields from the earlier Assets table must now be divided between

these two new tables. And, one record in the earlier Assets table now gives rise to two records (one in each of the new tables).

For this reason, you must create primary IDs in the Portfolio items table now. This is because these records must be unique throughout the entire AssetCenter database, and not just throughout one table.

You must create a Mapping element of the following type:

```
<Mapping to="amPortfolio" from="amAsset">
  <Field sqlname="lPortfolioItemId"
value="SDU_NEWID+lAstId"/>
</Mapping>
```

Transferring a character to a field

AssetCenter 4.1.0 enables you to access new fields whether they originate from the standard structure of the database or a customization that you made.

You might want to use one of these new fields instead of a feature used in the database to be converted.

This is only useful for the features used extensively.

Advantages

The fields can be positioned easier than the features in a detail window.

Disadvantages

- The **Available** field (seAvailable) of the feature parameters does not have an equivalent at the field level.
- Unlike features, the fields cannot be associated to classes.

Syntax

```
<Mapping to="[SQL name of the destination table]" from="[SQL
name of the source table that stores the feature values]">
  <Field sqlname="[SQL name of the destination field]"
value="[SQL name of the field that stores the feature
values]" feature="[SQL name of the source feature]"
```

```
featuretable="[SQL name of the table that stores the feature
values]" />
</Mapping>
```

You must use aliases for all the tables. These aliases are used at the attribute level, except in the case of the `value` attribute, which references the field that stores feature values.

The `Value` attribute can take the following values:

- **ValString** if the features stores text.
- **fVal** if the feature stores a number.
- **dtVal** if the feature stores a date.

Example

```
<Mapping to="amComputer A" from="amAsset">
  <Field sqlname="VideoCard" value="ValString"
feature="Video card" featuretable="amFVAsset" />
</Mapping>
```

Limitations

This methodology of transferring features to fields has a few limitations:

- It requires using numerous joins.
- It risks slowing the conversion performances.
- It does not enable you to manage feature heritages.
- It does not enable you to manage the deletion of transferred feature values.

You must add a `PostActions` element after the `Field` element to perform this task.

If you need to convert several features, it is preferable to use the `<Script>` element, such as in the following example:

```
UPDATE amComputer
SET ComputerDesc = (SELECT F.ValString
  FROM amFVAsset F, amFeature V, amAsset A
  WHERE lComputerId = SDU_NEWID * 2 + A.lAstId AND F.lFeatId
= V.lFeatId AND V.SQLName='fv_BiosMachine')
GO
```

```
DELETE FROM amFVAsset WHERE lFeatValId IN ( SELECT lFeatValId
FROM amFVAsset F, amFeature V WHERE F.lFeatId = V.lFeatId
AND V.SQLName='fv_BiosMachine' )
GO
```

Converting a field that stores application to be manually converted

The fields that store application data to be manually converted are purposely emptied during the conversion using the Mapping element. Here is such an example:

```
<Mapping to="amAccessRestr" from="amAccessRestr">
  <Field sqlname="ReadCond" value="'" />
</Mapping>
```

The records containing emptied application data are still conserved during the migration, though.

The application data to be manually converted is not lost. This is because it was exported with AssetCenter Database Administrator before the conversion, and it will be restored later during the conversion process.

Using joins

The joins must respect the following rules:

- An alias must be defined for each of the tables in the join.
- The expressions of the attributes where, orderby, groupby, having and value in the Field elements identify tables by their aliases.

Warning:

The joins concerning **long text fields** or **variable length binary fields** are not supported.

Example

```
<Mapping from="amProdSoftInfo s, amSoftware soft"
to="amCatProduct" where="s.lSoftId = soft.lSoftId">
  <Field sqlname="lCatProductId" value="s.lProdSoftId"/>
  <Field sqlname="InternalRef"
value="soft.Publisher+'/' +soft.Name+'/' +soft.VersionLevel"/>
  <Field sqlname="FullName"
value=" '/' +soft.Publisher+' ':' +soft.Name+' ':' +soft.VersionLevel+' '/'"/>
  <Field sqlname="dtLastModif" value="s.dtLastModif"/>
</Mapping>
```

Note:

The first table specified by a `from` attribute has a particular status.

This table's fields are automatically associated with the fields in the destination table having the same SQL name if they are not in the conversion file.

Populating foreign keys

The foreign keys are used to create links between the records of different tables.

Example

```
<Mapping from="amAsset" to="amPortfolio"
  <Field sqlname="lParentId" value="SDU_NEWID+lParentId"/>
</Mapping>
```

Dividing source tables between two or more destination tables

If you must divide a source table between two or more destination tables, you need to have a technique to make sure the primary IDs created in the destination tables will be unique throughout the AssetCenter database.

This technique involves creating a `Field` element of the type:

```
<Mapping to="amPortfolio" from="amAsset">
  <Field sqlname="lPortfolioItemId" value="SDU_NEWID * 2 +
lAstId" />
</Mapping>
```

Converting a numeric string into a text string

The conversion of data is sometimes necessary to convert a numeric string into a text string.

This is the case when you must calculate the value of a **Text** field according to a **Numeric** field, for example.

This is a complex conversion to carry out using an SQL language, and it varies from engine to engine.

We have created a SDUSTR macro that can easily handle this conversion for all engines and all types of numeric fields.

For example:

```
<Mapping to="amPortfolio" from="amSoftInstall">
  <Field sqlname="Code" value="'^' || SDUSTR lInstId"/>
</Mapping>
```

In this example:

- The `lInstId` field is a **32-bit integer number** type field.
- The `Code` field is a **text** type field.
- The `lInstId` is transformed into a text string by the SDUSTR macro.
- The converted string is concatenated with the `^` character.
- The concatenated string is inserted into the `Code` field.

Manually converting application data

The role of certain Mapping elements is to empty the application data to manually convert.

Here is such an example:


```
<Mapping to="amAccessRestr" from="amAccessRestr">
  <Field sqlname="TableName" value="'" />
</Mapping>
```

The emptied fields are populated again during the restoration of the application data that was manually converted.

SQL commands generated from a conversion file

The conversion file is used to generate SQL commands that the DBMS uses to modify the database (structure and data).

Example

The following Mapping element:

```
<Mapping from=[F] to=[T] where=[W]>
  <Field sqlname=[F1] value=[V1]/>
  <Field sqlname=[F2] value=[V2]/>
  ...
  <Field sqlname=[Fn] value=[Vn]/>
</Mapping>
```

Has as its SQL equivalent:

```
Insert Into to T(F1; F2, ..., Fn)
Select V1 as F1, V2 as F2, ..., Vn as Fn
From A
Where W
```

Verifying the conversion file before using it

Warning:

You must validate how the conversion file conforms to the **acmig.dtd** file before using it for the conversion.

To validate its conformity, you must use Internet Explorer or a text editor.

Here are some other tests that we recommend doing:

- The conversion file must not contain any occurrences of the combinations (`from`, `to`, `where`, `groupby`).
- The `Mapping` elements are in line with how you use the database.
- The multiple primary keys created from the same source primary key are different (appropriate use of the `SDU_NEWID` variable).
- The foreign keys that store primary keys created during the conversion correspond to the correct primary keys.
- The source and destination fields that are not associated (either manually in the conversion file or automatically by the conversion tool) are purposely unassociated.

To perform this verification:

- 1 Display the **`sdu.xml`** file (located in the conversion log folder).
- 2 Search for **`NotMappedSrc`** and **`NotMappedDst`**.

- The sub-set of records defined by the `where` attributes are not recovered.
They cover all the records.
- The tables associated several times do not trigger the creation of multiple links to the same record when such links can only exist once (**`lParentId`** or **`lCommentId`**, links for example).

Structural modification made to the database between versions

AssetCenter 4.1.0 is installed with files (**`diff*.*`**) that describe the differences between the structure of the standard database and:

- The version 4.1.0.
- A given earlier version.

 **Warning:**

The **`diff*.*`** files do not take into account the customizations you might have made to your previous database.

The **diff*.*** files are available in several formats:

- Text (**diff*.txt**).
- XML (**diff*.xml**).
- HTML (**diff*.htm**).

These files are usually located in the **C:\Program Files\Peregrine\AssetCenter\infos** folder.

They are installed if you select the **Documentation** package during the installation.

The name of these files is in the form:

diff<earlier version of AssetCenter>.*



Tip:

You will find these numbers by launching your previous version of AssetCenter and consulting the menu **Help/ About AssetCenter**.

Using diff*.txt files

Open these files under Excel or another tool specifying that the file is a DOS (or ASCII) format text file.



Tip:

Under Excel, we recommend applying an automatic filter to the first line in order to filter the information according to the changes you wish to see.

The header explains the contents of each column.

Each following line corresponds to modification to the database.

Here is some information about certain of the available columns:

- Name of the table containing the object:
 - Creation of table:

```
<SQL name in a new database>
```

- Deletion of table:

```
<SQL name, or, if it's not available, technical name
in the previous database>
```

- Creation, deletion or modification of field, index or link;
Modification of table:

```
<SQL name, or, if it's not available, technical name  
in the previous database> (<SQL name in the current  
database>)
```

- Name of the object that has been modified:

- Object destroyed:

```
<SQL name, or, if it's not available, technical name  
in the previous database>
```

- Object modified:

```
<SQL name, or, if it's not available, technical name  
in the previous database> (<SQL name in the current  
database>)
```

- Object added:

```
<SQL name in a new database>
```

- Description:
 - Object modified or created: new description of the object.
 - Object destroyed: previous description of the object.

Using the diff*.htm files



These files can be consulted using an HTML browser.


Here is the structure. You can search the following expressions to browse through these files.

1 Deleted table information

This title is at the beginning of each section that describes a deleted table.

For each table you will find:




- Information about the deleted table.
-  Fields of the deleted table.
-  Links of the deleted table.

-  Index of the deleted table.

2 Inserted table information

This title is at the beginning of each section that describes an added table.

For each table you will find:

- Information about the added table.
-  Fields of the added table.
-  Links of the added table.
-  Index of the added table.

3 Modified table

- Deleted objects
- Inserted objects
- Modified objects

Using the diff*.xml files

These files will come in handy if you are experienced in XML and have needs that require an XML file.

Examine these files yourself to determine what needs you might have for them.

Application data to be manually converted

This section contains the list of application data to verify during the migration.

Tip:

This application data needs to be verified because it references tables, fields or links that might have been deleted or modified in the version 4.1.0.

Application data stored in the AssetCenter Script Analyzer database

The different application data enter into one of the following categories:

- Basic script
- AQL query
- Field that stores the name of a table.
- Field that stores the name of a field.
- Wizard
- Calculated string (string of links and fields in a given context).

To learn how to verify and correct the data and parameters, refer to this guide's chapter **Step-by-step migration - simulating the conversion of the working database**, section **Processing application data to be manually converted**.

This application data is accessible using AssetCenter's graphical interface.

During the conversion, this data is not modified.

AssetCenter Script Analyzer analyzes the potential problems and enables you to manually modify the application data to be manually converted. This allows you to adapt the data to the structure of the version 4.1.0 database.

Table 8.1. Application data to be manually converted - list

Table (SQL name)	Field or link (SQL name)	Restrictions
amAction	WizardScript	
	Script	
	MsgTo	
	MsgCc	
	MsgBcc	
	Subject	
	memMsgText	

ActionFile		
Folder		
Parameters		
DDEService		
DDETopic		
DDECommand		
ContextTable		
RefObject		
Table (SQL name)	Field or link (SQL name)	Restrictions
amQuery		
	memQueryText	
	TableName	
amWfActivity		
	memScript	
	ContextTable	
amWfEvent		
	AQLCond	
	memScript	
	ContextTable	
	MonitTable	
	MonitFields	
amCalcField		
	memScript	
	Aql	Version 4.0.0 only
	ComputeString	Version 4.0.0 only
	TableName	
	Script	Version 4.0.0 only
amAccessRestr		
	WriteCond	
	TableName	
	ReadCond	
amTaxFormula		
	memFormula	
	TableName	
amWfOrgRole		
	memScript	
	ContextTable	
amFeatParam		
	AvailScript	
	DefValScript	

MandatScript		
ForceDspScript		
HistScript		
TableName		
LinkFilter		
Table (SQL name)	Field or link (SQL name)	Restrictions
amFeatScript		
	memScript	
amOption		
	memOptValue	
amFieldAdjustTempl		
	memScript	
	ContextTable	
	TargetField	
amFieldAdjust		
	TargetField	
	AdjustedTable	Version 4.0.0 only
amDeprScheme		
	memScript	
amLoan		
	ProrateField	
amCntrRent		
	ProrateField	
amDateAlarm		
	MonitoredField	
	MonitoredTable	
amLabelRule		
	memScript	Versions 3.1.0 et 4.0.0 only
	TableName	Versions 3.1.0 et 4.0.0 only
	FieldName	Versions 3.1.0 et 4.0.0 only
amCatRefScript		
	memScript	Version 4.0.0 only
amScriptLibrary		
	memScript	Version 4.0.0 only
amCbKStoredEvent		
	FieldName	Version 4.0.0 only
	Context	Version 4.0.0 only
amCbKRule		
	AmountField	Version 4.0.0 only
	EvtField	Version 4.0.0 only

Table (SQL name)	Context	Restrictions
amCbkScript		
	Context	Version 4.0.0 only
	memScript	Version 4.0.0 only

Other application data to verify

The following types of application data are neither converted during the database conversion nor verified using AssetCenter Script Analyzer:

- Help on fields
- Forms
- Views
- Import scripts
- Web pages of:
 - AssetCenter Web
 - Get-It
 - Get-Resources
- Connect-It scenarios
- AssetCenter Export export script
- Crystal Reports

These types of application data must be tested one by one.

Structural parameters of the database

These parameters are defined with AssetCenter Database Administrator. During the conversion, these parameters are propagated to the standard **gbase.dbb** file of the 4.1.0 database.

Table 8.2. Structural parameters of the database - list

Database object	Parameter	Available in versions:	
		3.0.1, 3.0.2, 3.1.0, 3.5.1, 3.5.2 et 3.6.0	4.0
Table	String	Yes	Yes
Table	Validity	Yes	Yes
Table	Relevance	No	Yes
Field or link	History	Yes	Yes
Field or link	Read only	Yes	Yes
Field or link	Mandatory	Yes	Yes
Field or link	Irrelevance	No	Yes
Field or link	Default value	Yes	Yes

Other documentation (Migration)

The **AssetCenter 4.1.0 - Migration** guide only provides information directly related to the migration.

To obtain associated information not covered in this guide, we recommend that you read the following documents:

Table 8.3. Other documentation (Migration) - list

Document	Information	Format	Location in the AssetCenter installation folder
Differences between the versions 3.x and 4.1.0	<ul style="list-style-type: none"> List of new features in the version 4.1.0 	Printed	<code>\doc\pdf\diff*.pdf</code>
		Online	<code>\doc\pdf\diff*.pdf</code>
Readme	<ul style="list-style-type: none"> Last-minute information 	Text	<code>readme.txt</code>
Release Notes	<ul style="list-style-type: none"> List of documentations provided with AssetCenter. Overview of new functions 	Printed	<code>\doc\pdf\relnotes*.pdf</code>
		Online	<code>\doc\pdf\relnotes*.pdf</code>

Document	Information	Format	Location in the AssetCenter installation folder
Installation	<ul style="list-style-type: none"> List of AssetCenter programs Supported operating systems and minimum configuration Supported DBMSs Installing AssetCenter 	Printed	\doc\pdf\installbook*.pdf
		Online	\doc\chm\installbook*.chm
Database structure	<ul style="list-style-type: none"> List of the database's tables, fields, links and indexes Automatic agents triggered by AssetCenter. 	Text file	<ul style="list-style-type: none"> \infos\database.txt \infos\tables.txt
		Printed	\doc\pdf\dbstruct*.pdf
		Online	\doc\chm\dbstruct*.chm
Differences between version 3.x and version 4.0.0	<ul style="list-style-type: none"> List of tables, fields, links and indexes that have changed. 	Text	\infos\diff*.txt
		HTML	\infos\diff*.html
Administration	<ul style="list-style-type: none"> AssetCenter Database Administrator Import 	Printed	\doc\pdf\admin*.pdf
		Online	\doc\chm\admin*.chm
Advanced use	<ul style="list-style-type: none"> Data export 	Printed	\doc\pdf\advan*.pdf
		Online	\doc\chm\advan*.chm

For more information about XML, consult the Web site:
<http://www.w3.org/XML/>.

