Peregrine

AssetCenter 4.3.2

Migration

Peregrine SYSTEMS

PART NUMBER DAC-432-EN23

© Copyright 2004 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.

If you need technical support for this product, or would like to request documentation for a product for which you are licensed, contact Peregrine Systems, Inc. Customer Support by email at support@peregrine.com.

If you have comments or suggestions about this documentation, contact Peregrine Systems, Inc. Technical Publications by email at doc_comments@peregrine.com.

This edition applies to version 4.3 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

Migration

Table of Contents

Introduction **11**

Why migrate? 12
What does migration involve? 12
Who is migration intended for? 13
Required competencies 13
How to use this guide 13

Chapter 1. Supported environments **15**

Operating systems and DBMSs 15
AssetCenter **working database** 15
Disk space required for the DBMS server 16

Chapter 2. Migration process **17**

Chapter 3. Step-by-step migration - preparation phase (working database) **29**

Preliminary analysis 29
Launching the migration project 30
Training the users and support technicians 31
Preparing your conversion computer 32
Preparing the DBMS server 34

Chapter 4. Step-by-step migration - simulation (simulation database) 35

Step 1 - Verify the integrity of the old-format working database 35
Step 2 - Manually adjust the old-format working database 37
Step 3 - Propagate structure changes made to the old-format working database 48
Step 4 - Copy the old-format working database 52
Step 5 - Convert the **old-format simulation database** 54
Step 6 - Verify the integrity of the **4.3.2-format simulation database** . . 67
Step 7 - Validate the **4.3.2-format simulation database**. 68
Step 8 - Restrict certain rights on the old-format working database . . . 69
Step 9 - Export application data to be manually converted 70
Processing application data to be manually converted 73
Step 14 - Adapt the integration with external tools 89

Chapter 5. Step-by-step migration - final conversion (migration database) 91

Step 15 - Verify the integrity of the old-format working database 91
Step 16 - Block and copy the old-format working database 92
Step 17 - Convert the **old-format migration database** 92
Step 18 - Restore the manually converted application data 93
Step 19 - Verify the integrity of the **4.3.2-format migration database** . . 93
Step 20 - Finalize the **4.3.2-format migration database** 93
Step 21 - Upgrade the external software components that access the AssetCenter database 108

Chapter 6. Step-by-step migration - final phase 111

Step 22 - Upgrade the AssetCenter programs 111
Step 23 - Put the **4.3.2 format migration database** into production . . . 115
Step 24 - Uninstall programs no longer used 116

Chapter 7. Glossary 117

Migration 117
Updating AssetCenter programs 117
Converting the old-format working database 118
Conversion file 118

Conversion machine 118
Working database 119
Trigger 119
Data 119
Application data 119
Database structure 120

Chapter 8. References 121

Adapting the **migration.xml** conversion file 121
Structural modifications to the standard database compared to previous versions 140
Application data to be manually converted 142
Structural parameters from the **old-format working database** propagated . 147
Other documentation 148

Index 151

List of Figures

2.1. Conversion - Process to convert AssetCenter version 4.1.x or earlier 22
2.2. Conversion - process to convert AssetCenter version 4.2.x 25
4.1. Propagating structural changes - process 49
4.2. Processing sample data - procedure 72
4.3. AssetCenter Script Analyzer - ***.xml** file analysis window 80
4.4. AssetCenter Script Analyzer - script analysis window 83

List of Tables

1. Operations to perform depending on the previous version number 12
4.1. Fields that must not contain the ^ character - list 40
4.2. Fields that must not contain the / character - list 42
4.3. AssetCenter Script Analyzer - menus 78
8.1. Application data to be manually converted - list 143
8.2. Structural parameters of the **old-format working database** - list 147
8.3. Other documentation - list 148

PREFACE

Introduction

The operations to be performed to upgrade from a previous version of AssetCenter to version 4.3.2 depend on the number of your previous version.

These simple cases are refered to as **simple upgrades** and are described in the **Installation and Upgrade** guide, chapter **Upgrading a previous version of AssetCenter**.

Important:

If you fall into the case of a **simple upgrade**, the **Migration** guide will not concern you.

The more complex cases are refered to as **migrations** (full or simplified, depending on the previous version number) and are described in this guide.

The following table enables you to determine which case you are in.

Table 1. Operations to perform depending on the previous version number

AssetCenter previous version number	Operations to perform	Relevant documentation
Version 4.3.0 or later	Simple upgrade	**Installation and Upgrade** guide, chapter **Upgrading a previous version of AssetCenter**
Versions 4.2.x	In these standard situations, a simple upgrade will suffice.	**Installation and Upgrade** guide, chapter **Upgrading a previous version of AssetCenter**
	If the simple upgrade fails, you will need to perform a simplified migration process.	**Migration** guide
Versions 4.1.x or earlier	Full migration	**Migration** guide

Why migrate?

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

- The standard database structure (tables, fields, links, indexes) has been vastly modified.
- New functions have been added.

All these changes have made it necessary to methodically migrate your earlier version of AssetCenter to the 4.3.2 version.

What does migration involve?

Migration involves performing the following tasks:

- Converting the old-format working database to the 4.3.2 format (structure and content).
- Upgrading the AssetCenter programs to the version 4.3.2.

Who is migration intended for?

This migration is performed by the engineers in charge of:

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

Required competencies

Migration is a complex process that requires:

- A thorough understanding of the earlier versions of AssetCenter and of version 4.3.2 (installation, configuration of parameters, database structure, functions, administration, interfacing with external applications).
- Preparation
- Technical competency: SQL, database administration.
- Methodology
- Time
- Resources

How to use this guide

Tip:

Before reading this guide, we recommend that you read some of the other AssetCenter 4.3.2 guides:

- **Installation**
- **Release Notes**
- **readme.txt**
- **Differences between the versions 3.x and 4.x**

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 Migration process

This chapter provides an overview of the migration process.

This process will differ depending on which version of AssetCenter you are migrating.

Read the sections in the chapter corresponding to your previous installed version to learn about the steps in the migration process.

Chapter Step-by-step migration - preparation phase (working database)

Chapter Step-by-step migration - simulation (simulation database)

Chapter Step-by-step migration - final conversion (migration database)

Chapter Step-by-step migration - final phase

These chapters describe each step in the migration process.

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, paying attention to each detail.

Chapter Glossary

This chapter defines the key terms used in migration.

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

Chapter References

This chapter contains exhaustive and systematic reference information.

Read this chapter to obtain advanced or supplementary information.

CHAPTER 1 Supported environments

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 working database

This migration supports the conversion of the following databases:

- AssetCenter version 3.01 and later and included Service Packs.

  If the format of your **working database** is earlier than the version 3.0.1, you must first convert your **working database** to the 3.0.2 format.

  To learn how to convert a database to the 3.0.2 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 4.3.2.

Disk space required for the DBMS server

Warning:

If you are migrating a version 4.1.0 or later of AssetCenter, you can skip this section.

The disk space allotted by the DBMS server to the **old-format working database** must be at least twice the size of the **old-format working database**.

CHAPTER 2

Migration process

What does migration entail?

This migration is a set of operations required to convert an earlier version of AssetCenter to the version 4.3.2:

- Converting the **old-format working database** (structure and contents) in order to make it compatible with the 4.3.2 version of AssetCenter.
- Updating the AssetCenter programs to the version 4.3.2 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:

- Making the structure of the current database conform to that of the 4.3.2 version of AssetCenter.
- Conserving original data whenever possible.

- Modifying the data that cannot be conserved in its original state 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.

  For a list of these data items: ▶ Application data to be manually converted [page 142].

Warning:

The conversion tools can only be used to for migration.

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

Which process should I use to migrate from AssetCenter version 4.1.x or earlier?

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

1 Simulate the conversion on the **simulation database**:

  Verify the integrity of the **old-format working database** using AssetCenter Database Administrator.

  ▶ Step 1 - Verify the integrity of the old-format working database [page 35]

  Manually adjust the **old-format working database** using AssetCenter.

  This prepares the **old-format working database** in order so that is can be converted.

  ▶ Step 2 - Manually adjust the old-format working database [page 37]

  Propagate the structural changes you made to the **old-format working database** to the **standard 4.3.2 gbbase*.*** database-description files.

  ▶ Step 3 - Propagate structure changes made to the old-format working database [page 48]

4 Make a backup of the **old-format working database**. This backup will be called the **simulation database**. While you simulate the conversion on this **simulation database**, the users continue to work on the **old-format working database**.

▸ Step 4 - Copy the old-format working database [page 52]

5 Convert the **old-format simulation database** using AssetCenter Database Administrator. Adapt and test the **migration.xml** conversion file if necessary.

▸ Step 5 - Convert the old-format simulation database [page 54]

6 Verify the integrity of the **4.3.2-format simulation database** using AssetCenter Database Administrator.

▸ Step 6 - Verify the integrity of the 4.3.2-format simulation database [page 67]

This enables you to make sure the conversion has not corrupted the **simulation database**.

7 Validate the **4.3.2 format simulation database**.

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

▸ Step 7 - Validate the 4.3.2-format simulation database. [page 68]

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

▸ Step 8 - Restrict certain rights on the old-format working database [page 69]

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

▸ Step 9 - Export application data to be manually converted [page 70]

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

▸ Step 10 - Verify and correct the application data [page 74]

11 Restore the manually converted application data in the **4.3.2-format simulation database**. Do this using AssetCenter Script Analyzer or AssetCenter Database Administrator.

▸ Step 11 - Restore corrected application data [page 85]

12 Verify the integrity of the **4.3.2-format simulation database** using AssetCenter Database Administrator.

This enables you to make sure the restoration has not corrupted the **base de simulation**.

▸ Step 12 - Verify the integrity of 4.3.2-format simulation database [page 87]

13 Test the restored application data using AssetCenter 4.3.2.

▸ Step 13 - Verify restored application data [page 87]

14 Prepare for the adaptation of the AssetCenter 4.3.2 integration with external applications.

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

▸ Step 14 - Adapt the integration with external tools [page 89]

2 Convert a second backup of the old-format working database called the **migration database**:

15 Verify the integrity of the old-format working database using AssetCenter Database Administrator.

▸ Step 15 - Verify the integrity of the old-format working database [page 91]

16 Block the old-format working database and make a backup called the **migration database**.

▸ Step 16 - Block and copy the old-format working database [page 92]

17 Convert the **old-format migration database** using AssetCenter Database Administrator.

▸ Step 17 - Convert the old-format migration database [page 92]

18 Restore the manually converted application data in the **4.3.2-format migration database**. Do this using AssetCenter Script Analyzer or AssetCenter Database Administrator.

▸ Step 18 - Restore the manually converted application data [page 93]

19 Verify the integrity of the **4.3.2-format migration database** using AssetCenter Database Administrator.

▸ Step 19 - Verify the integrity of the 4.3.2-format migration database [page 93]

20 Finalize the **4.3.2-format migration database** using AssetCenter to finish the conversion. It is this **4.3.2-format migration database** that you will put into production after upgrading the programs.

- Step 20 - Finalize the 4.3.2-format migration database [page 93]

3 Upgrade the external software components that access the AssetCenter database.

- Step 21 - Upgrade the external software components that access the AssetCenter database [page 108]

4 Upgrade the AssetCenter programs.

- Step 22 - Upgrade the AssetCenter programs [page 111]

5 Put the **4.3.2 format migration database** into production.

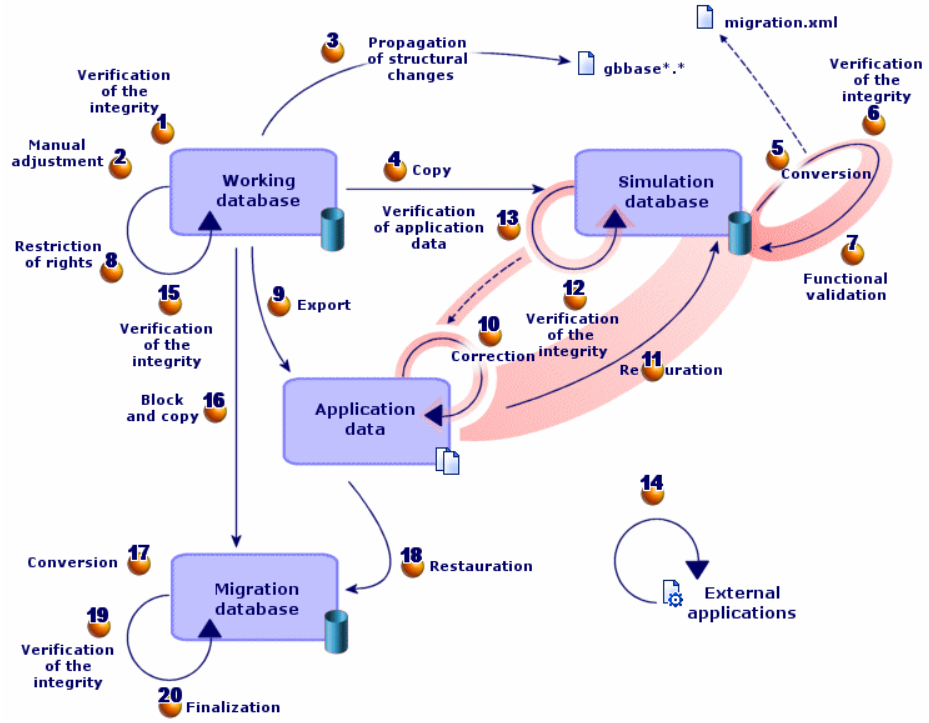
- Step 23 - Put the 4.3.2 format migration database into production [page 115]

6 Uninstall any programs no longer used.

- Step 24 - Uninstall programs no longer used [page 116]

Here are the main steps in the conversion process:

Figure 2.1. Conversion - Process to convert AssetCenter version 4.1.x or earlier



Note:

Why does certain application data need to be converted manually?:

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.

For a list of these data items and parameters: ▸ Application data to be manually converted [page 142].

Which process should I use to migrate from AssetCenter version 4.2.x?

The process is a simplified version of the migration process for AssetCenter version 4.1.x or earlier:

1 Simulate the conversion on the **simulation database**:

Verify the integrity of the **old-format working database** using AssetCenter Database Administrator.

▸ Step 1 - Verify the integrity of the old-format working database [page 35]

If necessary, manually adjust the **old-format working database** using AssetCenter.

▸ Step 2 - Manually adjust the old-format working database [page 37]

Propagate the structural changes you made to the **old-format working database** to the **standard 4.3.2 gbbase*.*** database-description files.

▸ Step 3 - Propagate structure changes made to the old-format working database [page 48]

Make a backup of the **old-format working database**. This backup will be called the **simulation database**. While you simulate the conversion on this **simulation database**, the users continue to work on the **old-format working database**.

▸ Step 4 - Copy the old-format working database [page 52]

Convert the **old-format simulation database** using AssetCenter Database Administrator.

This converts the structural parameters of the **old-format simulation database** while preserving any structural changes you might have made.

▸ Convert the old-format simulation database [page 55]

Verify the integrity of the **4.3.2-format simulation database** using AssetCenter Database Administrator.

This enables you to make sure the conversion has not corrupted the **simulation database**.

▸ Step 6 - Verify the integrity of the 4.3.2-format simulation database [page 67]

2 Convert a second backup of the old-format working database called the **migration database**:

7 Block the old-format working database and make a backup called the **migration database**.

▶ Step 16 - Block and copy the old-format working database [page 92]

8 Convert the **old-format migration database** using AssetCenter Database Administrator.

▶ Convert the old-format simulation database [page 55]

Instead of connecting to the **old-format simulation database**, you must connect to the **old-format migration database**.

9 Verify the integrity of the **4.3.2-format migration database** using AssetCenter Database Administrator.

▶ Step 6 - Verify the integrity of the 4.3.2-format simulation database [page 67]

Instead of connecting to the **old-format working database**, connect to the **4.3.2-format migration database**.

3 Finalize the **4.3.2 format migration database** using AssetCenter to finish the conversion process. It is this **4.3.2 format migration database** that you will put into production after upgrading the programs.

▶ Step 20 - Finalize the 4.3.2-format migration database [page 93]

4 10 Upgrade the external software components that access the AssetCenter database.

▶ Step 21 - Upgrade the external software components that access the AssetCenter database [page 108]

5 11 Update the AssetCenter programs.

▶ Step 22 - Upgrade the AssetCenter programs [page 111]

6 12 Put the **4.3.2 format migration database** into production.

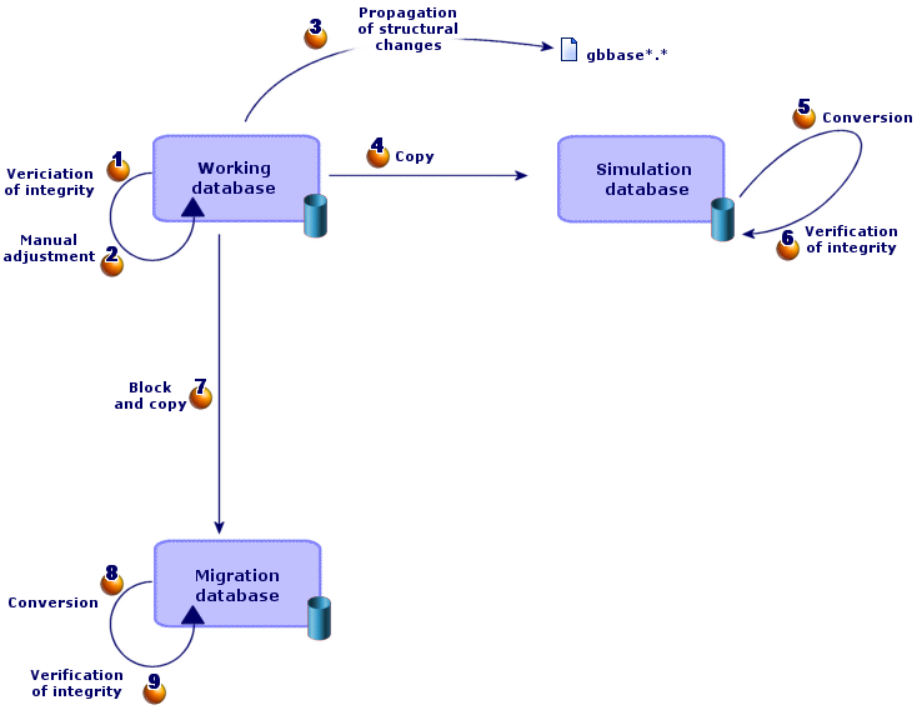
▶ Step 23 - Put the 4.3.2 format migration database into production [page 115]

7 13 Uninstall any programs no longer used.

▶ Step 24 - Uninstall programs no longer used [page 116]

Here are the main steps in the conversion process:

Figure 2.2. Conversion - process to convert AssetCenter version 4.2.x



How do the conversion tools work?

The conversion tools are integrated into:

- AssetCenter Database Administrator 4.3.2.
- AssetCenter Script Analyzer 4.3.2.

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 made to the old-format working database to the **standard 4.3.2 gbbase*.*** database-description files.

- **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 old-format working 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 old-format working 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 into 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.

Migrating a database whose DBMS is not supported by version 4.3.2

If the DBMS of the old-format working database is not supported by version 4.3.2:

1 Transfer the old-format working database to a DBMS that is supported by AssetCenter 4.3.2.

  To learn how to do this, refer to the **Administration** 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 old-format working 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 old-format 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 old-format working database.

CHAPTER 3 Step-by-step migration - preparation phase (working database)

This chapter explains step-by-step which operations to perform before converting.

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 4.3.2.

   ▸ Other documentation [page 148]
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 documentation for users and their training.

Launching the migration project

Taking into account the extent of the improvements and changes made to the version 4.3.2 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:

Out of prudence, certain enterprises prefer splitting up the migration process into several separate phases:

1 Starting out by obtaining the functional equivalent of the previous version of AssetCenter and stabilizing this.

2 Exploring the new functionality in AssetCenter 4.3.2.

This will ensure a smoother transition.

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 old-format working 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 before you can put the **4.3.2-format working database** 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 the old-format 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 old-format working database

You need to install this version to access the working databases:

- **Working database**
- **Simulation database**
- **Migration database**

At the least, you must install the basic module.

Verifying you have access to the old-format working database

You need to do this in order to:

- Prepare the old-format working database for the conversion.
- Make a backup of the old-format working database to simulate and then perform the conversion.

Installing AssetCenter 4.3.2

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 4.3.2)

This is required to restore the application data to be converted manually after correction.

Note:

You will need an authorization key to use Connect-It. Make sure as soon as possible that you have this key. If necessary, contact Peregrine Systems to obtain a key before you need to use Connect-It.

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 4.3.2)

You will need this tool to convert the customizations made to the structure of the old-format working database.

Factors affecting the conversion rate

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

Tip:

If you have a large volume of data in the old-format working database, you must position the computers where AssetCenter Database Administrator is installed as close as possible to the databases (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 old-format databases

During the migration, you will have to convert the **old-format simulation database** and the **old-format migration database**.

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

▸ Disk space required for the DBMS server [page 16]

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.

CHAPTER 4 Step-by-step migration - simulation (simulation database)

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

These simulations cannot be performed on the **working database**, though. They can only be done on a **simulation database**.

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

After the simulations are complete, you can convert another backup of the old-format working database, called the **migration database**.

It is this **4.3.2-format migration database** that will be put into production.

This chapter explains step-by-step which operations to perform on the **simulation database**.

Step 1 - Verify the integrity of the old-format working database

1

Important:

Make a backup of the old-format working database.

2 Perform an initial verification with the old-version AssetCenter Database Administrator:
   1 Launch the old-version AssetCenter Database Administrator.
   2 Connect to the **old-format working database** (**File/ Open** menu, **Open existing database** option).
   3 Display the database-diagnostics window (**Action/ Diagnostics / Repair database** menu).
   4 Select (**No table**) in the list of tables. In reality, this corresponds to (**All tables**).
   5 Specify the name and the location of the log file.
   6 Only select the **Checking validity of records** option.
   7 Select the **Repair** option.
   8 Click **Start**.
   9 Consult the messages of the execution window.
   10 Consult the log file if necessary.

   Tip:

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

3

Warning:

If the DBMS of the **old-format working database** is DB2, stop here without performing the second verification.

Perform a second verification with 4.3.2-format AssetCenter Database Administrator:
   1 Launch AssetCenter Database Administrator 4.3.2.
   2 Connect to the **old-format working database** (**File/ Open** menu, **Open existing database** option).

   Note:

   It is fully possible to connect to the previous format database using AssetCenter Database Administrator 4.3.2.

   3 Display the database-diagnostics window (**Action/ Diagnostics / Repair database** menu).
   4 Select (**All tables**) in the list of tables.

5 Specify the name and the location of the log file.
6 Select all the verification options, except for the **Checking validity of records** option.
7 Select the **Repair** option.
8 Click **Start**.
9 Consult the messages of the execution window.
10 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 **Administration** guide, chapter **Diagnostic and repairs of a database**.

Step 2 - Manually adjust the old-format working database

Warning:

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

Certain data must be modified before converting the **old-format working database** in order that the process is carried out smoothly.

Most of the constraints to respect in the **old-format working database** 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.

Adjustments concerning all versions of the old-format working database

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 (for AssetCenter versions 3.x)

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 **old-format working database**, you must:

1. Make a copy of the **up_GetCounterVal** stored procedure if you plan on modifying it in the same way after converting.
2. Manually update the counters in the **amCounter** table that were diverted to other tables.
3. Restore the stored procedure **up_GetCounterVal** to its original state.

Tip:

You will reapply the directives in the technical notes in the step Step 20 - Finalize the 4.3.2-format migration database [page 93].

Mandatory nature of fields and links

Certain fields and links need to be populated before a record can be created in a given table.

The mandatory nature of the fields and links is defined either in the AssetCenter database or in the **gbbase*.*** database-description files.

This mandatory nature can either be true in all cases, or it can be calculated with a script.

The records created or modified by the conversion program must respect the mandatory nature of the fields and links. This mandatory nature is stated in the **customized 4.3.2 gbbase*.*** database-description files.

Fields and links 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).

The **migration.xml** conversion files installed by default with AssetCenter 4.3.2 are intended to work properly when the format of the **old-format working database** and the database-description files, **standard 4.3.2 gbbase*.***, have not been modified.

The standard **migration.xml** conversion files cannot be adapted except for in the following cases:

- If ever you deleted the mandatory nature of a field or link during your use of the old-format working database.
- If you added the mandatory nature to certain fields or links of the **standard 4.3.2 gbbase*.*** database-description files.

To populate the mandatory fields and links, the conversion file might use certain data from the **old-format working database**.

You must make sure that the fields and links that are declared mandatory in the **customized 4.3.2 gbbase*.*** database-description files are populated in the old-format working database before the conversion.

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 old-format working database are used to populate other fields in the 4.3.2-format working 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 **customized 4.3.2 gbbase*.*** files).

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 old-format 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 old-format working database):

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

SQL name of the table	SQL name of the field
amProduct	Model
amProduct	CatalogRef
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
amBudget	Type
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

SQL name of the table	SQL name of the field
amLocation	FullName
amLocation	Name
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
amTaxFormula	Code

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 **lLicCntrId** foreign key (in the **amAsset** table).

Warning:

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

Full name fields

When you use a character string containing the / character to populate a **Full name** field, the / character is interpreted as a hierarchical-level separator.

For certain DBMSs, this is not a problem because the standard conversion files could be configured to replace the / characters with a neutral character.

If you use Sybase Adaptive Server, you cannot make such a replacement on the fly.

You must instead replace the / character by another character of your choice in the fields used to populate a **Full name** field.

You should verify these fields:

Table 4.2. Fields that must not contain the / character - list

SQL name of the table	SQL name of the field	Versions concerned:						
		3.0.1	3.0.2	3.1.0	3.5.0	3.5.1	3.6.0	4.0.0
amItemListVal	Value (for brands)	Yes	Yes	Yes	Yes	Yes	Yes	
amFamily	Brand	Yes	Yes	Yes	Yes	Yes	Yes	
amFamily	Name	Yes	Yes	Yes	Yes	Yes	Yes	
amAsset	ComputerName						Yes	
amAsset	AssetTag	Yes	Yes	Yes	Yes	Yes	Yes	
amInvoice	InvoiceNumber	Yes	Yes	Yes	Yes	Yes	Yes	
amProduct	Model	Yes	Yes	Yes	Yes	Yes	Yes	
amSoftware	Publisher	Yes	Yes	Yes	Yes	Yes	Yes	

SQL name of the table	SQL name of the field	Versions concerned:						
		3.0.1	3.0.2	3.1.0	3.5.0	3.5.1	3.6.0	4.0.0
amSoftware	Name	Yes	Yes	Yes	Yes	Yes	Yes	
amSoftware	VersionLevel	Yes	Yes	Yes	Yes	Yes	Yes	
amContract	Ref	Yes	Yes	Yes	Yes	Yes	Yes	
amAdjustment	ItemNo	Yes	Yes	Yes	Yes	Yes	Yes	
amConsUse	ItemNo	Yes	Yes	Yes	Yes	Yes	Yes	
amComputer	Name							Yes

Functional domains

Note:

This section only deals with AssetCenter versions 4.0.0 and earlier.

The **Name** field of the records in the **amFuncDomain** table must respect the constraints of the SQL names: Only letters of the English alphabet, numbers and the "_" character are authorized. This is because these fields will be used to populate the **SQL name** field in the 4.3.2 database.

Adjustment concerning only versions 3.6.0 and earlier of the old-format working database

Itemized-list values

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

Elementary adjustments

The records in the **Elementary adjustments** table (amFieldAdjust) whose **Adjustment** link is not populated disappear during the conversion.

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

This is why you must make sure before you convert that all records you want to convert respect this conversion.

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 section Rules used for the old-format simulation database whose versions are earlier than 4.0.0 [page 60].

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 **lLicCntrId** 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).

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

  ▸ Adapting the migration.xml conversion file [page 121]

  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.

Estimates

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 **lPOrdLineId** 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) 4.3.2 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 **lPOrdLineId** 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 most 9 levels.

To respect this condition, move the product packages whose **sLvl** field is superior or equal to **9** up in the hierarchy.

Tip:

You can also modify the script of the conversion file so that is manages more levels.

1 Search the following pair of lines in the <**PostActions**> element corresponding to your DBMS:

```
UPDATE amCatProduct SET FullName = Q.FullName || amCatProduct.Inter
nalRef || '/', sLvl = Q.sLvl + 1 FROM amCatProduct, amCatProduct Q
WHERE amCatProduct.sLvl = -1 AND Q.lCatProductId = amCatProduct.lPa
rentId AND Q.sLvl <> -1
GO
```

The number of times this pair of lines appears is the number of supported levels.

2 Add one pair per additional hierarchic level you want the file to support.

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**),
- And 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 most 10 levels.

To respect this condition, move the order lines whose **sLvl** field is superior or equal to **10** up in the hierarchy.

Tip:

You can also modify the script of the conversion file so that is manages more levels.

1 Search the following pair of lines in the <**PostActions**> element corresponding to your DBMS:

```
UPDATE amPOrdLine SET FullName = Q.FullName || amPOrdLine.ItemNo ||
 '/', sLvl = Q.sLvl + 1 FROM amPOrdLine, amPOrdLine Q WHERE amPOrdL
ine.sLvl = -1 AND Q.lPOrdLineId = amPOrdLine.lParentId AND Q.sLvl <
> -1
GO
```

The number of times this pair of lines appears is the number of supported levels.

2 Add one pair per additional hierarchic level you want the file to support.

Categories

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

To respect this condition, move the categories whose **sLvl** field is superior or equal to **10** up in the hierarchy.

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.

Step 3 - Propagate structure changes made to the old-format working database

Warning:

To perform this operation, the **standard 4.3.2 gbbase*.*** database-description files that you use here must be the standard file installed with AssetCenter 4.3.2. You cannot use these files if any modifications were made

This operation:

- Concerns the users who modified the standard structure of the old-format working database (addition or modification of fields, indexes and tables) and want to keep those changes in the 4.3.2-format working database.
- Aims to propagate the structural modifications in the **standard 4.3.2 gbbase*.*** files.

  Tip:

  The **standard 4.3.2 gbbase*.*** files obtained will be used to structure the **4.3.2-format database** during the conversion.

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

Warning:

Only the structural changes made to the old-format working database using AssetCenter Database Administrator will be accounted for.

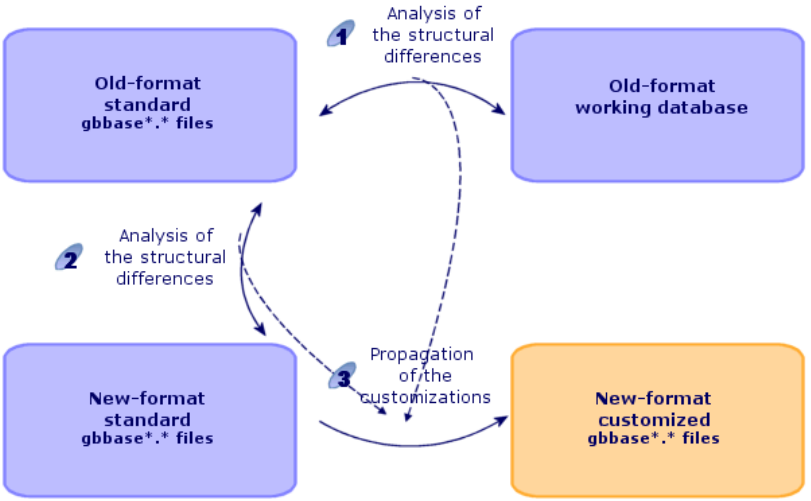
You must manually delete all structural changes made by any other means in the old-format working database.

List of propagated structural parameters: ▶ Structural parameters from the old-format working database propagated [page 147].

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 old-format working database and the **standard old-format gbbase*.*** files.

2: The tool determines the differences between the **standard old-format gbbase*.*** files and the **standard 4.3.2 gbbase*.*** files.

3: The tool copies and modifies the **standard 4.3.2 gbbase *.*** files 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 4.3.2 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 will then appear.

  Note:

  Exception: If a **Name** or **Description** modification is detected for the same table, field or link in the steps 1 and 2, it is the modification detected at step 1 that is applied.

- In the **old-format working database** - before definitively propagating the structural changes - you must modify the SQL names of tables, fields and indexes that appear in the version 4.3.2.

  Otherwise they will conflict with the standard field of the version 4.3.2 with the same name.

The **customized 4.3.2 gbbase*.*** files must be clearly identified. This will come in handy in the following steps:

- Step 9 - Export application data to be manually converted [page 70]
- Processing application data to be manually converted [page 73]
- Step 5 - Convert the old-format simulation database [page 54]
- Step 17 - Convert the old-format migration database [page 92]

Propagate the structural changes.

1. Launch AssetCenter Database Administrator 4.3.2.
2. Connect to the old-format working database with the **Admin** login (**File/ Open/ Open existing database** menu).
3. Select the **Migration/ Propagate the customized structure** menu.
4. Follow the instructions given by the wizard.
5. 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.

6 If the messages tell you so, modify the structure of the old-format working database. Then perform the migration starting from the step Step 4 - Copy the old-format working database [page 52].

This must be repeated until you obtain a good **customized 4.3.2 gbbase*.*** files without any problem messages.

7

Warning:

This step does not concern you if you are migrating a version 4.3.0 or later of AssetCenter.

Certain scripts might not be propagated to the **standard 4.3.2 gbbase*.*** files.

There will be a message in the **newdbb.log** log file and an **.xml** file created in the **<Generation folder>\dbbscript** and **<Generation folder>\builddbb\dbbscripts** folders for each script that is not propagated.

These customizations must be propagated manually in the **customized 4.3.2 gbbase*.*** files.

You can go to step Step 10 - Verify and correct the application data [page 74] 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 **customized 4.3.2 gbbase*.*** files using AssetCenter Database Administrator.

8 If you are converting an old-format working database whose version is later or equivalent to 4.0.0, 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 4.3.2 gbbase*.*** files again when you execute the step Step 5 - Convert the old-format simulation database [page 54].

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.

Anaylze and adapt the migration.xml conversion file to handle structural changes

Warning:

This does not concern you if you are migrating a version 4.3.0 or later of AssetCenter.

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.

Step 4 - Copy the old-format working database

Problems that can occur during a traditional backup

If you make a backup of the old-format working database using DBMS tools, the backup of the old-format working database 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 old-format working 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.
- Make a backup of the old-format working database in an empty database using AssetCenter Database Administrator.

Note:

The backup of the old-format working 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 old-format working database using the DBMS tools

1 Copy the old-format working database using the DBMS tools.

The backup is identical to the original old-format working database.

2 Delete all the modifications made to:
   - Indexes
   - Triggers
   - Stored procedures
   - Views

3 Create an AssetCenter connection to **old-format simulation database**.

Solution 2: Copy the old-format working database into an empty database using AssetCenter Database Administrator

1 Create an empty, old-format AssetCenter database.

2 Create an AssetCenter connection to this empty database.

3 Open the **old-format working database** in AssetCenter Database Administrator.

4 Copy the **old-format working database** into the previously created empty database (**Action/ Copy database to empty database** menu).

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

To learn how to make a backup of the old-format working database in an empty database using AssetCenter Database Administrator, refer to the **Administration** guide, chapter **Using a test database**, section **Copying your working database**.

Step 5 - Convert the old-format simulation database

Warning:

The conversion tools must not be used to modify the structure of the **4.3.2-format working database** (adding, deleting or modifying tables, fields, indexes, stored procedures, triggers, views, etc.).

Such modifications must be planned after the migration.

Adapt 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 4.3.2 is installed with conversion files by default (1 file per earlier version of AssetCenter that is supported by the migration).

These files describe what data to transform during the conversion of the **old-format simulation database**, as well as 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 **simulation database** before being executed on the **migration database** in a later step.

Syntax of the conversion files and how to customize them: ▶ Adapting the migration.xml conversion file [page 121].

Constraints caused by modifying the data in the **old-format working database**: ▶ Step 2 - Manually adjust the old-format working database [page 37].

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.

Convert the old-format simulation database

To convert the **old-format simulation database**:

1. If the DBMS of the **working database** is DB2, execute the SQL commands that are in the file: **db2udb72.sql**. This file is usually located in the folder: **C:\Program Files\Peregrine\AssetCenter\migration\db2udb72** sur la **base de simulation ancien format**.
2. Launch AssetCenter Database Administrator version 4.3.2.
3. Connect to the **old-format simulation database** with the **Admin** login (**File/ Open/ Open existing database** menu).

Important:

In the connection detail 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).
- With Microsoft SQL Server, if the owner of the tables is **dbo**, the connection login must create default tables in the form **dbo.<table>** (typically the login: **sa**).

4 Select **Migration/ Convert the database**.

5 Follow the instructions given by the wizard.

Tip:

Converting fields whose **User type** is **Comment** takes a lot time (several hours for a large database).

Because no messages appear during this phase, you might be wondering if the conversion process is still running.

To make sure, examine the activity on the conversion machine or on the database server (CPU or I/O).

6 Consult the messages of the **sdu.log** log file.

Warning:

This does not concern you if you are migrating a version 4.3.0 or later of AssetCenter.

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

1 Correct the source of the problem.

2 Restart the conversion from step Step 4 - Copy the old-format working database [page 52].

Information about the conversion

Warning:

This section does not concern you if you are migrating a version 4.3.0 or later of AssetCenter.

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 old-format simulation database

Floor plan positions

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

Structural parameters of the database

The conversion program applies all the parameters of the tables, fields, links and indexes defined in the selected **customized 4.3.2 gbbase*.*** database-description files.

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 **old-format simulation database**.

You might want to populate the information necessary in order for the mandatory fields be populated. This information should go into the old-format 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.

Tip:

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

Index of unique values

The conversion file does not systematically 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 4.3.2?	Are there modifications to fields, links or indexes between the old version and version 4.3.2?	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 using the orders defined in the description file.

The **migration.xml** conversion files installed by default are written so that the emptied fields correspond to exported application data.

Rules used for the old-format simulation database whose versions are the same as or earlier than 4.0.0

System data

AssetCenter is provided with a set of data that you can import into a demonstration database or into your working database:

- System data: basic but indispensable data for the AssetCenter application to function.

  System data has not been specifically identified until the version 4.0.0..

  This data cannot be modified by the user.
- Line-of-business data: basic data to insert into your working database if you want to.

  This data is divided into functional groups.
- Sample data: data useful for familiarizing yourself with AssetCenter.

During conversion, the system data of the old-format working database is automatically and integrally replaced by the system data of the version 4.3.2.

Rules used for the old-format simulation database whose versions are earlier than 4.0.0

Natures

Natures are created from asset categories.

The names of natures do not necessarily have any significance.

Categories having the same properties (**Nature** field (seNature), for example) create one, single nature with corresponding properties.

History

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

Assets

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

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

If the **old-format simulation database** is version 3.5.0 or earlier: If a feature containing information of the same nature is associated with the transferred asset - and if this feature is populated - then the value of the feature overrides the value obtained by transferring 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)
- **lCPUCount** (equivalent to the **lCPUNumber** 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 **4.3.2-format simulation 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 arbitrarily 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 is of the product is different than **0**.
- The product is linked to a record in the **amProdSupp**, **amPOrdLine**, **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 transferred to the **amCatRef** table if the products are linked to a record in the **amProdSupp**, **amPOrdLine**, **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.

Asset features

Asset features are attached to the portfolio item associated with the asset when it is converted. This is true except for when a feature is transferred to a 4.3.2 version database field (in particular fields of the **Computers** table).

In the case where a feature is transferred to a field, the feature is unattached from the asset without being attached to the associated portfolio item.

Tip:

The conversion files contain `Mapping` elements that you can activate to modify the behavior that we just described.

Estimate

The estimates that created a totally or partially received order disappear during the conversion.

The other estimates are transformed into orders.

Potential sources of conflict

Identifiers

During the conversion, the new IDs (primary keys) are created for each record created in a table.

However, the number of IDs is limited to 2^31 at the database level, no matter what DBMS you are using.

If this number is exceeded, the final database will be corrupt.

No error messages will warn you of this during the conversion.

You must therefore verify yourself before the conversion that this number has not been exceeded.

The maximum number of IDs created during the conversion depends on the version of the **old-format simulation database**.

To verify that this number has not been exceeded:

1 Determine the approximate value of the largest ID (let's call it **MaxId**) in the **old-format simulation database**.

To do this, create a record in any table (**amLocation**, for example). Note the value of this new record's primary key (**lLocaId** for the **amLocation** table).

Tip:

To view this value, just add this field to the list: Right-click and select **Configure list** from the contextual menu.

2 Verify that **MaxId** is less than (2^31)/8.

Note:

There are no constraints if the **old-format simulation database** is a version **4.0.0**.

Order-line brand

The value of the **Brand** field (Brand) from the order lines linked to a product (**Product** link) is lost during the conversion. This is because the product itself is linked to a brand.

The value of the **Brand** field (Brand) from the other order lines is added to the **Description** field (LineDesc).

Request-line brand

The value of the **Brand** field (Brand) from the request lines linked to a product (**Product** link) is lost during the conversion. This is because the product itself is linked to a brand.

The value of the **Brand** field (Brand) from the other request lines is added to the **Description** field (LineDesc).

Unique indexes

Unique indexes have been added to certain tables in version 4.3.0 of AssetCenter.

Note:

These new unique indexes provide you with reliable reconciliation keys where previously unavailable.

An example of their usefulness: When exporting data from AssetCenter to be modified outside of AssetCenter then reimported in AssetCenter. Using the reconciliation key, the previous records can be located and updated without any duplicate records being created.

Consequences: Certain uniqueness constraints might not be respected in the old-format database.

Whenever this happens, the database conversions is interrupted.

The conversion program warns you and provides a list of conflicts.

Follow the instructions given by the conversion program.

Products whose brands and models are the same, but which the categories are different

Products such as these cannot be converted.

Whenever this happens, the database conversions is interrupted.

The conversion program warns you and provides a list of conflicts.

Follow the instructions given by the conversion program.

Step 6 - Verify the integrity of the 4.3.2-format simulation database

1 Important:

Make a backup of the old-format working database.

2 Launch AssetCenter Database Administrator 4.3.2.

3 Connect to the **4.3.2-format simulation 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, except for the **Checking validity of records** option.
8 Select the **Analyze only** 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.

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 Step 5 - Convert the old-format simulation database [page 54].

Warning:

The two previous operations do not concern those users that migrate a version 4.1.0 or later of AssetCenter.

Or:

1 Modify the data in the old-format working database.
2 Start again from step Step 4 - Copy the old-format working database [page 52].

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

7 Step 7 - Validate the 4.3.2-format simulation database.

Browse the **4.3.2-format simulation database** to see if the conversion appears correct.

You can notably:

- Compare the number of records found in the main tables between the **4.3.2-format simulation database** and the **old-format simulation database**

If there is too big a difference in number, verify that this is normal.

Example of a big - but normal - difference: The license contracts are deleted from the Contracts table during the conversion. It is thus normal that the number of records in the Contracts table greatly diminishes.

- Examine the detail of at least one record in each main table to see if the information is coherent.

  With contracts, especially, you should examine at least one record per contract type (lease, maintenance, etc.).

  You should pay particular attention to the sensetive links, such as the **Model** link at the asset level.
- Make sure the features and their values were correctly converted and that the conversion of features into fields has gone smoothly (one test per feature).

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

1 Modify the **migration.xml** conversion file.

2 Start again from step Step 5 - Convert the old-format simulation database [page 54].

Or:

1 Modify the data in the old-format working database.

2 Start again from step Step 4 - Copy the old-format working database [page 52].

Step 8 - Restrict certain rights on the old-format working database

Modify the user rights of the old-format 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 converted manually: ▶ Application data to be manually converted [page 142].

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 old-format working database. The modifications made to the backup of the working database are not recovered in the conversion process.

Step 9 - Export application data to be manually converted

Reminder

▸ Application data to be manually converted [page 142]

Tip

There is probably a large amount of application data.

You may consider deleting obsolete application data from the **previous format database** before exporting the application data.

This will save you from having to test the any migrated application data that would be obsolete.

Export the application data to be manually converted

1. Launch AssetCenter Database Administrator 4.3.2.
2. Connect to the old-format working database with the **Admin** login (**File/ Open/ Open existing database** menu).
3. Select the **Migration/ Export application data** menu.
4. Follow the instructions given by the wizard.
5. 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.

6. 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 old-format working database.
- Excludes the system data.

  This data is processed in a specific manner, which is described in section Information about the conversion [page 57].
- Do not verify if the tables, links and fields of the application data conform to the structure of the 4.3.2 database.

  Tip:

  This is done by the AssetCenter Script Analyzer.
- Includes the **line-of-business data** and the **sample data**.

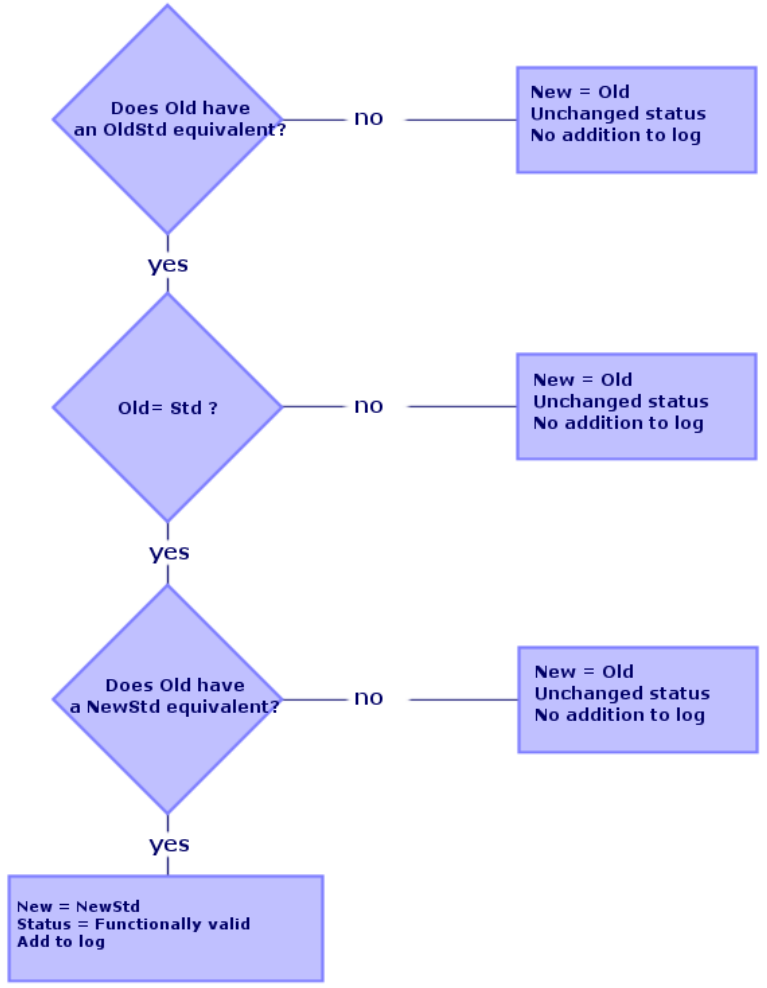
The **sample data** is processed in a specific manner.

The objective of this specific process is to automatically update the unchanged, sample application data in your old-format working database.

To process this data, the tool examines each item of application data that you exported, one by one.

Here is the procedure:

Figure 4.2. Processing sample data - procedure



Definition:

- **Old**: an exported application data item (in other words, a data item from your old-format working database).
- **OldStd**: if any old-format standard **sample data** exists, it corresponds to **Old**.

The reconciliation between **Old** and **OldStd** is done using an ID that depends on the type of data. For an action, for example, this would be the **SQL name** field.

The sample application data that the tool uses as its reference is stored in the AssetCenter 4.3.2 installation folder. It is located in the **\migration\fromXxx\reference** sub-folder where **Xxx** corresponds to the number of the old version of AssetCenter.

- **NewStd**: if any 4.3.2-format standard **sample data** exists, it corresponds to **Old**.

  The reconciliation between **Old** and **NewStd** is done using an ID that depends on the type of data.

  The sample application data that the tool uses as its reference is stored in the AssetCenter 4.3.2 installation folder. It is located in the **\migration\fromXxx\referencenew** sub-folder where **Xxx** corresponds to the number of the old version of AssetCenter.
- **New**: **Old** after it is processed by the tool (either modified or left as it is).

Processing application data to be manually converted

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

1 Step 10 - Verify and correct the application data [page 74]
2 Step 11 - Restore corrected application data [page 85]
3 Step 12 - Verify the integrity of 4.3.2-format simulation database [page 87]
4 Step 13 - Verify restored application data [page 87]

These steps are described in this section.

Note:

In this section, when we mention **fields** needing to be verified and possibly replaced with new values, we are referring to fields and links in the structure of the AssetCenter database.

Tip:

You can divide the processing of application data among several people, but it must be managed as a coherent project.

Step 10 - Verify and correct the application data

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 below.

3 If you have created a tree structure of **.xml** files containing scripts that are not propagated in the step Step 3 - Propagate structure changes made to the old-format working database [page 48]:

1 Copy the **<Generation folder>\dbbscript** and **<Generation folder>\builddbb\dbbscript** folders created in the step Step 3 - Propagate structure changes made to the old-format working database [page 48] (if they exist).

2 Propagating structural changes: ▶ Step 3 - Propagate structure changes made to the old-format working database [page 48].

3 Paste this folder into the folder specified by the **Working folder** field.

4 Display the list of application data to be verified (via the **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.

When exporting application data, AssetCenter Database Administrator automatically assigns an SQL name to the **.xml** files. By default, this name is composed of a prefix followed by an automatically incremented number.

In some cases, you can use a more precise name:

SQL name of the table	SQL name of the field used to give a name to the .xml file.
amAction	SQLName
amQuery	SQLName
amCalcField	SQLName
ItemNo	amFieldAdjustTempl

SQL name of the table	SQL name of the field used to give a name to the .xml file.
ItemNo	amFieldAdjust
OptSection	amOption

5 At this stage, if you want to process the scripts that have not been propagated automatically in step Step 3 - Propagate structure changes made to the old-format working database [page 48], start with the **.xml** files corresponding to these scripts:

1 Select the first **.xml** file from the **<Generation folder>\dbbscript** and **<Generation folder>\builddbb\dbbscript** folders.

2 Analyze the file in detail (via the **Actions/List the problems in the script** menu).

3 Consult the **Message** window.

See 13 and 14.

4 Use the modification suggestions proposed by AssetCenter Script Analyzer to modify the corresponding scripts in the **customized 4.3.2 gbbase*.*** files obtained in step Step 3 - Propagate structure changes made to the old-format working database [page 48].

For this, launch AssetCenter Database Administrator and open the **customized 4.3.2 gbbase.xml** file. Then perform the script modifications by hand.

5 When you finish processing the **.xml** file, select the **Functionally valid** option.

6 Display the listvia the **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 script in its context** menu).

The purpose of this operation is to verify that the script is valid according to the version 4.3.2 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.

Warning:

Testing the script in its context does not exempt you from testing the script using the **Actions/List the problems in the script** menu: The tool tests the script's different aspects.

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 **4.3.2-format simulation database** to test the application data that was manually converted.

When you try to select the **Restorable** option, the script is automatically tested in its context.

See 3.

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.

Note:

Certain application data already have the **Functionally valid** status.

This data is the application data found among the sample data in the step Rules to respect during the export [page 71].

Speeding up the correction of application data

Warning:

The tip that we provide here is a rather delicate procedure.

You should not perform this procedure unless you are perfectly comfortable with this task, and you are fully responsible for the actions.

Certain of these corrections will seem repetitive.

So you can perform Search and Replaces on the set of **.xml** files.

Here are some precautions to take:

- Make a backup of the **.xml** files after each step before doing the Search and Replace.
- Include a delimiter in the string you search.
- Select the **Whole word** option for the search.
- Request a count of the number of replacements and make sure that number seems right.
- Analyze the differences at the level of the modified files before and after the replacement.

AssetCenter Script Analyzer menus

Table 4.3. AssetCenter Script Analyzer - menus

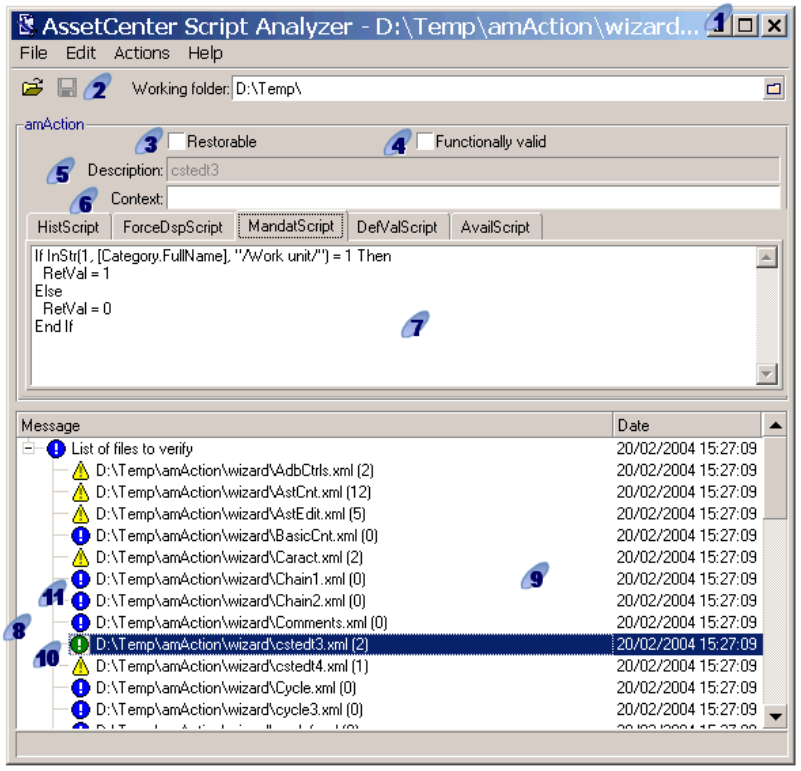
Menu	Use
File menu	
New	Not needed.
Open	Enables you to open an **.xml** file from the tree structure whose root is defined by the **Working folder** field.
Save	Saves the modifications made to the file (**Restorable** or **Functionally valid** character, context, scripts).
Save as	Not needed.
Exit	Exits 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.

Menu	Use
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. Warning: Only use this menu when: • Using the menu **Actions/ Validate the script in its context** returns an unjustified error. • You are certain of the script's validity. Example of its usefulness: The **Schedule level rents** table (amCntrRent) contains a **Prorated by** field (ProrateField). This field stores the **System name** of a field. However, AssetCenter Script Analyzer only knows how to detect inconsistencies in SQL names. AssetCenter Script Analyzer displays an unjustified error in most cases. For this field, do the following: 1 Select the **Actions/ Force the restorable nature of the file** menu. 2 Restore the file. 3 Test the file in AssetCenter 4.3.2. 4 Correct the value of the **Prorated by** field in AssetCenter Script Analyzer. 5 Select the **Functionally valid** option.
List unprocessed files	Displays the list of **.xml** files: • 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.
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 4.3. AssetCenter Script Analyzer - *.xml file analysis window



1 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 is 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 4.3.2, 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.

8 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).

9 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 ℹ, 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.

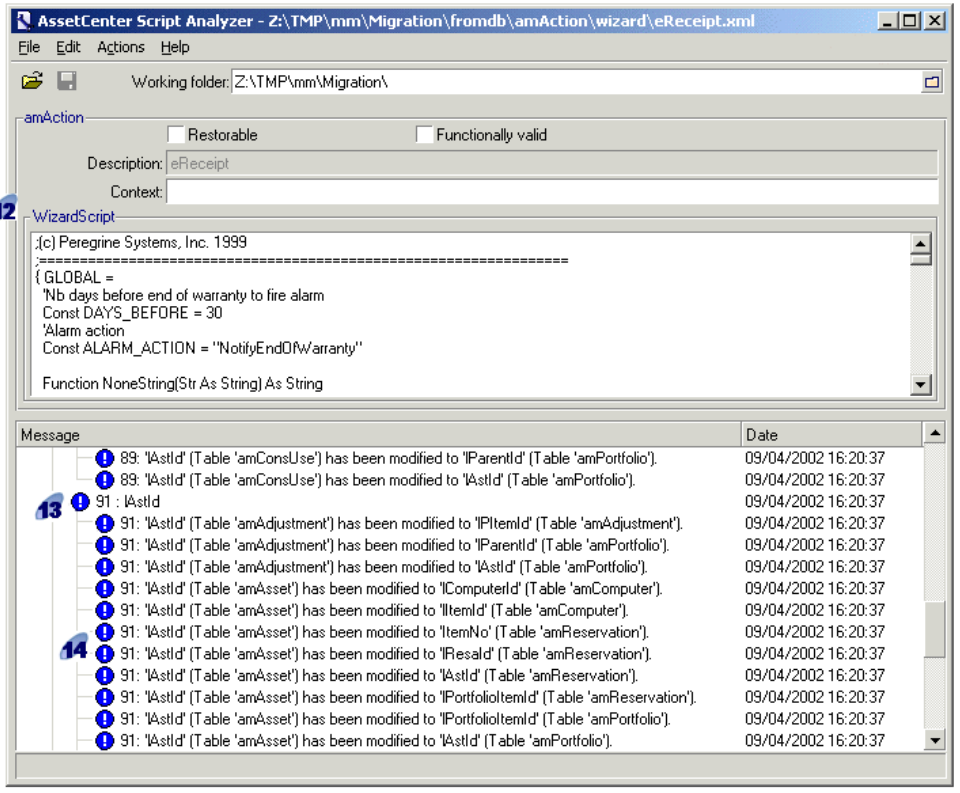
10 The green icon indicates that the **.xml** file is **Functionally valid**.

11 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 4.4. AssetCenter Script Analyzer - script analysis window



Note:

The **Message** window only analyzes the current script.

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 **customized 4.3.2 gbbase*.*** target database description files.

  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 **customized 4.3.2 gbbase*.*** target database description files.
- `'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 **customized 4.3.2 gbbase*.*** target database description files. 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.

Step 11 - 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.

Restore application data corrected with AssetCenter Database Administrator

1. Launch AssetCenter Database Administrator 4.3.2.
2. Connect to the **4.3.2-format simulation database** with the **Admin** login (**File/ Open/ Open existing database** menu).
3. Select the **Migration/ Restore the application data** menu.
4. Follow the instructions given by the wizard.
5. 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.

Restore 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 **4.3.2-format simulation 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

- The application data stored in an **.xml** file that is declared non-restored is rejected.

- Any mandatory field in the version 4.3.2 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.

Tip:

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

Step 12 - Verify the integrity of 4.3.2-format simulation database

Verify the integrity of the **4.3.2-format simulation database** as indicated in the section Step 6 - Verify the integrity of the 4.3.2-format simulation database [page 67].

Instead of connecting to the **old-format working database**, you must connect to the **4.3.2-format simulation 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.

Step 13 - Verify restored application data

Process

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 application data in the **4.3.2-format simulation 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.)

Note:

The end-of-paragraph characters are replaced with |.

This does not create any problems during the execution of the script.

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 **4.3.2-format migration database**.

Pitfalls

Concatenation operator

Certain versions of AssetCenter tolerated the + character as a string concatenation operator.

This character is now interpreted more strictly as an addition operator in version 4.3.2.

This may raise an error in AssetCenter when testing scripts.

In this case, replace the + operator with **&**.

Queries

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.

Step 14 - Adapt the integration with external tools

If you integrated external applications with the old-format working database, you will probably have to adapt the integration mode of these applications.

Potentially concerned applications: ▸ sections:

- AssetCenter Web [page 108]
- Get-It [page 108]
- Get-Resources [page 108]
- Connect-It scenarios [page 109]
- Import scripts [page 109]
- Export scripts [page 110]

You only implement the new integration mode in these applications after the step Step 20 - Finalize the 4.3.2-format migration database [page 93].

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

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

CHAPTER 5 Step-by-step migration - final conversion (migration database)

At this stage, you have:

- A set of **customized 4.3.2 gbbase*.*** files.

  ▶ Step 3 - Propagate structure changes made to the old-format working database [page 48]
- A **migration.xml** conversion file that was tested on the **simulation database**.
- Manually converted application data that was tested in the **4.3.2-format simulation database**.

This chapter explains step-by-step which operations to perform to convert the **4.3.2-format working database**.

15 Step 15 - Verify the integrity of the old-format working database

Verify the integrity of the old-format working database as indicated in the section Step 1 - Verify the integrity of the old-format working database [page 35].

Step 16 - Block and copy the old-format working database

Blocking the old-format working 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 old-format working database.

2 Shut down the:

- AssetCenter Server
- AssetCenter APIs
- External programs that access the old-format working database.

3 Block access to the old-format working database.

4 Make a backup of the old-format working database as described in the section Step 4 - Copy the old-format working database [page 52].

This backup of the old-format working database is called the **migration database**:

You need to minimize the time the old-format working database is blocked 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.

Step 17 - Convert the old-format migration database

To convert the **old-format migration database**, follow the instructions described in the section Convert the old-format simulation database [page 55]:

- Instead of connecting to the **old-format simulation database**, you must connect to the **old-format migration database**.
- You use the **migration.xml** conversion file that you finalized on the **simulation database**.

The actual conversion of the old-format migration database should be as brief as possible, because the old-format working databaseis 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 **old-format migration database**.

2 Put this blocked old-format working database back into production.
3 Redo simulations with a new **old-format simulation database**.
4 Perform the migration process again, starting up from the step Step 16 - Block and copy the old-format working database [page 92].

Step 18 - Restore the manually converted application data

To restore the application data that was manually converted in the **4.3.2-format migration database**, follow the instructions described in the section Step 11 - Restore corrected application data [page 85]:

- Instead of connecting to the **4.3.2-format simulation database**, you must connect to the **4.3.2-format migration database**.
- You use the **.xml** files in the working folder that you corrected using the **4.3.2-format simulation database**.

Step 19 - Verify the integrity of the 4.3.2-format migration database

Verify the integrity of the **4.3.2-format migration database** as indicated in the section Step 6 - Verify the integrity of the 4.3.2-format simulation database [page 67].

Instead of connecting to the **old-format working database**, connect to the **4.3.2-format migration database**.

Step 20 - Finalize the 4.3.2-format migration database

You will need to make alterations to the **4.3.2-format migration database** for several reasons:

- Certain data will not have been converted by the conversion program.

  You must test and manually alter certain data in the **4.3.2-format migration 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 **4.3.2-format migration database**.

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

Finalizations concerning all versions of the old-format production 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 **4.3.2-format migration 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 old-format working database.

Before converting the old-format working 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 (for AssetCenter versions 3.x)

Triggers, indexes, stored procedures and views

Before the conversion, you put the old-format 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 conversion of the **old-format migration database**, 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 4.3.2.
   2. Connect to the **4.3.2-format migration 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 **Administration** guide, chapter **Creating an AssetCenter API database**, section **Creating an empty AssetCenter API 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 4.3.2.

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 **standard 4.3.2 gbbase.xml** file, located in the **config** sub-folder of the AssetCenter 4.3.2 installation folder.
5 Select the **Action/ Create database** menu.
6 Select the **4.3.2-format migration 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 4.3.2, 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.

This 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 ^

Customizing the 4.3.2-format migration database and the detail of screens

In version 4.3.2, AssetCenter offers the possibility to add tables, fields, links, buttons and tabs.

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

Note:

If you have customized the +, - and **Magnifer** buttons, the customizations will not be maintained by the conversion.

Must must therefore modify again the buttons that you have customized.

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

Forms

When you convert the **old-format migration database**, the forms are left as they are.

The changes to the database structure are not recovered.

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

Test each form one at a time:

1. Launch AssetCenter 4.3.2.
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 **old-format migration 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 conversion of the **old-format migration database**, 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 4.3.2:

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 4.3.2

To import the **sample data** reports in the **4.3.2-format migration 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 **standard 4.3.2 gbbase.xml** file, located in the **config** sub-folder of the AssetCenter 4.3.2 installation folder.
5 Select the **Action/ Create database** menu.
6 Select the **4.3.2-format migration 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.

Finalizations concerning versions 3.6.0 and earlier of the old-format production database

Output events

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

Their values may reflect the structure of the old-format working database.

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.

Features replaced by a field

The conversion tool copied the values of these features to a target field.

On the other hand, unless you used scripts in the conversion file to delete these features and their values, you will have to do it manually.

Links that replace link-type features

When you transfer the values of the link-type features to a link in the 4.3.2 database, the link is not populated if its target table has changed during the conversion.

Example: Before the conversion, the feature points to the Assets table. After the conversion, the link replacing the feature points to the Portfolios table.

In this example, the ID of the asset disappears, and an ID is created for the new portfolio item.

After the conversion, you must execute a query to identify the records whose links (the ones that replaced the features) are not populated.

You must then populate these links manually.

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 **amCountry** 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**.

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

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

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 **4.3.2-format migration database**.

You can:

1. Identify the employees in the old-format working 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**).

▶ Rules used for the old-format simulation database whose versions are earlier than 4.0.0 [page 60]

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 4.3.2 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 that 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 onward, 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)**.

Step 21 - Upgrade the external software components that access the AssetCenter database

AssetCenter Web

You must upgrade AssetCenter Web to version 4.3.2.

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 4.3.2.

3 Test and adapt each Web page one after the other.

Get-It

For each Web application developed with Get-It to function with the AssetCenter 4.3.2 database:

1 Verify that your version of Get-It is listed in the AssetCenter 4.3.2 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 4.3.2 database:

1 Verify that your version of Get-Resources is listed in the AssetCenter 4.3.2 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 **4.3.2-format migration database** using Connect-It, you must use the version of Connect-It provided with AssetCenter 4.3.2.

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 4.3.2.
2 Connect to a test database (which can be a backup of your **4.3.2-format migration 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 4.3.2.
2 Connect to the **4.3.2-format migration 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.

CHAPTER 6 Step-by-step migration - final phase

This chapter explains step-by-step which operations to perform to get your **4.3.2-format migration database** up and running.

22 Step 22 - Upgrade the 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 4.3.2. 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 4.3.2, 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 **Administration** guide, chapter **AssetCenter Server**.

Delete the AssetCenter caches of the 4.3.2-format migration database

If you use a cache with the connection to your **4.3.2-format migration database**, we recommend that you delete this cache.

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

Upgrade AssetCenter programs

To upgrade the programs, install AssetCenter 4.3.2.

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 4.3.2 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 SQL Anywhere 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.

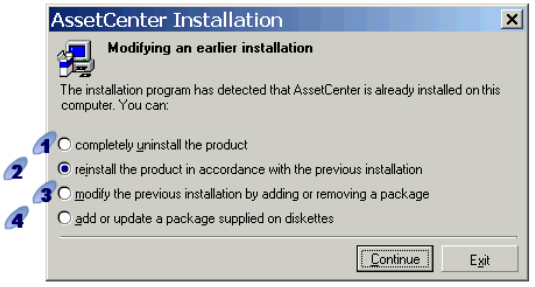
Tip:

If you are installing AssetCenter 4.3.2 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 4.3.2 of AssetCenter is already installed:

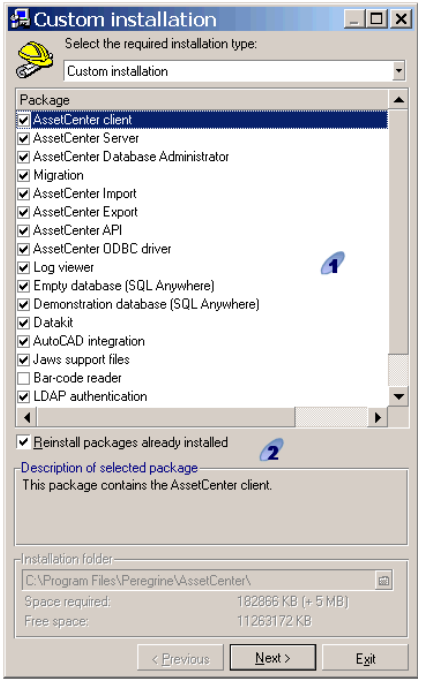


1 and 4: These options are not to be used during the upgrade.

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 4.3.2, contact user support.

Remove the old connections to databases and create new ones

The objective is to have the users connect to the **4.3.2-format migration database**.

Refer to the **User interface** 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 **User interface** guide, chapter **Customizing a client machine**.

Step 23 - Put the 4.3.2 format migration database into production

This is the last step of the migration process.

You have already:

- Totally converted the **old-format migration database** and fine tuned the **4.3.2-format migration 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 finalized **4.3.2-format migration database**.
2. Relaunch the external programs that access the **4.3.2-format migration database**.
3. Inform users that they can use the database.

Step 24 - Uninstall programs no longer used

At the start of the migration process, you will have installed certain programs on the conversion computer (▶ Preparing your conversion computer [page 148]).

We recommend keeping the following software for a period of time after the conversion on the conversion computer:

- The version of AssetCenter corresponding to the **previous format production database**: This will enable you to browse the **previous format production database**, if necessary, to verify data items before conversion.

You can uninstall the following programs from the conversion computer:

- Connect-It
- XML file editor
- Sun Java Runtime

By and large, once the conversion process is finished, the conversion computer and software are no longer required for the day-to-day running of AssetCenter 4.3.2 and the production database.

7 Glossary

CHAPTER

Migration

The migration is a set of operations required to convert an earlier version of AssetCenter to the version 4.3.2:

Migration includes:

- Converting the old-format working database (structure and contents) in order to make it compatible with the 4.3.2 version of AssetCenter.
- Updating the AssetCenter programs to the 4.3.2 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 4.3.2.

Do not confuse with ...

▸ Converting the old-format working database [page 118]

Converting the old-format working database

One of the operations required by the AssetCenter migration.

Converting the old-format working database involves modifying its structure and contents in order to make it compatible with the 4.3.2 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 [page 117]

Conversion file

A conversion file is a file that describes which data to transform during the conversion of the old-format working database 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 old-format working database to the 4.3.2 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.

Data

Data is information from a record that is stored in the database using a field.

Application data

Application data designates data that is stored in the working database (and not in its structure). It is this data that you must verify during the conversion.

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 4.3.2.

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).

Database structure

The database structure assembles the following items:

- Tables
- Fields
- Links
- Index

As well as their parameters, such as:

- Description string
- Validity
- Relevance
- History
- Read only
- Mandatory
- Irrelevance
- Default value

These parameters are defined by a fixed value or a script with AssetCenter Database Administrator.

They are stored in **gbbase*.*** database-description files or in the database itself.

8 References

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 4.3.2 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

Definition of a conversion file: ▶ Conversion file [page 118].

To learn when a conversion file needs to be adapted: ▶ Adapt the migration.xml conversion file [page 54].

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 **4.3.2-format database**.

Note:

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

The conversion file is used to generate SQL commands for modifying an **old-format 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 4.3.2 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]" featu
retable="[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 conversion of an **old-format database** (and even before you rename the previous tables).

Whenever possible, we recommend that you use a `PreActions` element. This element facilitates the maintenance of the conversion file.

A `StartScript` element is useful when:

- Several `Mapping` elements need to execute the same `PreActions` element.
- You need to remove the customizations made to the **old-format database** structure.
- You need to deactivate triggers.

The script must be written in an SQL language conforming to the one used in the DBMS of the **old-format 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 lItemListId=(SELECT lItemListId FROM a
mItemizedList 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.DocObjTab
le 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 4.3.2 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" wh
ere="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.

A `PreActions` element is useful when you:

- Create natures that are independent of the contents of the database to convert.
- Create a feature.

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 old-format working database.

`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 **4.3.2-format 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 **4.3.2-format 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 **old-format database** 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.

A PostActions element is useful when you:

- Calculate the value of the **Full name** field.
- Delete the features and feature values when they are transferred to a field.

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 old-format working database.

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.

Whenever possible, we recommend that you use a PostActions element. This element facilitates the maintenance of the conversion file.

A Script element is useful when:

- Several Mapping elements need to execute the same PostActions element.
- You want to perform clean-up operations that cannot be done using the Mapping element.
- You delete obsolete enumerations.

This element's syntax is the same as for a StartScript 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 old-format working database.

! - - 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 documentation.

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="$lt;'"
>	Closes an XML tag.	`</Mapping>`	$gt;	value=">'"
&	Marks the beginning of an entity.	`<`	&	value="&'"
;	Marks the end of an entity.	`<`	; without & before	value="';'"
\	SQL escape character.		\\	value="'\\'"

Special character	Interpretation	Example	Equivalent when the character must be interpreted as text.	Example
||	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 feature to a field

AssetCenter 4.3.2 enables you to access new fields whether they originate from the standard structure of the 4.3.2-format database or a customization that you made.

You might want to use one of these new fields instead of a feature used in the old-format working database.

This is only useful for the features used extensively.

Advantages

- The fields can be positioned easier than the features in a detail window.
- The access restrictions perform better on the fields than on the features.

Tip:

The access restrictions perform equally on the links as on the features.

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 t
he source table that stores the feature values]">
  <Field sqlname="[SQL name of the destination field]" value="[SQL nam
e 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" f
eaturetable="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 nor those of the features themselves.

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

  Otherwise, you can perform the deletion manually after the conversion.

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.lFeat
Id AND V.SQLName='fv_BiosMachine')
GO
DELETE FROM amFVAsset WHERE lFeatValId IN ( SELECT lFeatValId FROM amF
VAsset F, amFeature V WHERE F.lFeatId = V.lFeatId AND V.SQLName='fv_Bi
osMachine' )
GO
```

Potential problem with link-type features

When you transfer the values of the link-type features to a link in the 4.3.2 database, the link is not populated if its target table has changed during the conversion.

Example: Before the conversion, the feature points to the Assets table. After the conversion, the link replacing the feature points to the Portfolios table.

In this example, the ID of the asset disappears, and an ID is created for the new portfolio item.

After the conversion, you must execute a query to identify the records whose links (the ones that replaced the features) are not populated.

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.

The fields that store the names of tables not used as contexts for the elements to manually convert are automatically converted.

The conversion mechanism can be configured using a `Translate` element.

Using joins

Joins must respect the following rules:

- An alias must be defined for each of the tables of the join.
- Expressions of `where`, `orderby`, `groupby`, `having` and `value` attribute of `Field` elements must identify tables by their aliases.

Warning:

The joins concerning **Integer (32 bit)** or **variable length binary fields** are not supported.

Example

```
<Mapping from="amProdSoftInfo s, amSoftware soft" to="amCatProduct" wh
ere="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.Na
me+':'+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 **Number** 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 **Integer (32 bit)** 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 **old-format 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).

Transferring assets who don't have specific tables

In certain cases, there are **overflow tables** that enable you to describe certain types of specific assets (computers, for example).

In this case, `Mapping` elements have been added to the conversion files.

If there is no specific **overflow** table (for vehicles, for example), we recommend that you do not change data organization as it is in your old-format working database.

Structural modifications to the standard database compared to previous versions

AssetCenter 4.3.2 is installed with files (**diff*.***) that describe the differences in the database structures between:

- Version 4.3.2.
- A given previous version.

Warning:

The **diff*.*** files do not take into account any customizations you might have made to the **old-format working database**.

The **diff*.*** files are available in several formats:

- Text (**diff*.txt**).
- XML (**diff*.xml**).
- HTML (**diff*.htm**).

These files are generally located in the **C:\Program Files\Peregrine\AssetCenter\doc\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 can find the version number by launching the old versin of AssetCenter and opening the **Help/About AssetCenter** menu.

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 be able to filter the information according to the changes you wish to see.

The heading explains the contents of each column.

Each line after the heading corresponds to a structural modification in the standard database.

Here is some information about certain of the available columns:

- Name of the table containing the object:
  - Creation of table:

```
<SQL name in the 4.3.2-format standard database>
```

  - Deletion of table:

```
<SQL name, or if that doesn't exist, technical name in the old-f
ormat standard database>
```

  - Creation, deletion or modification of field, index or link; Modification of table:

```
<SQL name, or if that doesn't exist, technical name in the old-f
ormat standard database> (<SQL name in the 4.3.2-format standard
 database>)
```

- Name of the object that has been modified:
  - Object destroyed:

```
<SQL name, or if that doesn't exist, technical name in the old-f
ormat standard database>
```

  - Object modified:

```
<SQL name, or if that doesn't exist, technical name in the old-f
ormat standard database> (<SQL name in the 4.3.2-format standard
 database>)
```

  - Object added:

```
<SQL name in the 4.3.2-format standard 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 4.3.2.

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 verify and correct data and parameters: ▶ Processing application data to be manually converted [page 73].

This application data is accessible via 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 4.3.2 database.

Table 8.1. Application data to be manually converted - list

Table (SQL name)	Field or link (SQL name)	Restrictions
amAction		

Table (SQL name)	Field or link (SQL name)	Restrictions
	WizardScript	
	Script	
	MsgTo	
	MsgCc	
	MsgBcc	
	Subject	
	memMsgText	
	ActionFile	
	Folder	
	Parameters	
	DDEService	
	DDETopic	
	DDECommand	
	ContextTable	
	RefObject	
amQuery		
	memQueryText	
	TableName	
amWfActivity		
	memScript	
	ContextTable	
amWfTransition		
	AQLCond	
	LinkToTargetCtxTbl	
	TargetContextTable	
amWfEvent		
	AQLCond	
	memScript	
	ContextTable	
	MonitTable	
	MonitFields	
	LinkToMonitTable	
amCalcField		
	memScript	
	AQL	Starting from version 4.0.0.
	ComputeString	Starting from version 4.0.0.
	TableName	
	Script	Starting from version 4.0.0.
amAccessRestr		

Table (SQL name)	Field or link (SQL name)	Restrictions
	WriteCond	
	TableName	
	ReadCond	
amTaxFormula		
	memFormula	
	TableName	
amWfOrgRole		
	memScript	
	ContextTable	
amFeatParam		
	AvailScript	
	DefValScript	
	MandatScript	
	ForceDspScript	
	HistScript	
	TableName	
	LinkFilter	
amFeatScript		
	memScript	
amOption		
	memOptValue	
amFieldAdjustTempl		
	memScript	
	ContextTable	
	TargetField	
amFieldAdjust		
	TargetField	
	AdjustedTable	Starting from version 4.0.0.
amDeprScheme		
	memScript	
amLoan		
	ProrateField	
amCntrRent		
	ProrateField	
	Description	Starting from version 3.6.0.
amDateAlarm		
	MonitoredField	
	MonitoredTable	
amLabelRule		

Table (SQL name)	Field or link (SQL name)	Restrictions
	memScript	Versions 3.10, 4.0.0 and later
	TableName	Versions 3.10, 4.0.0 and later
	FieldName	Versions 3.10, 4.0.0 and later
amCatRefScript		
	memScript	Starting from version 4.0.0.
amScriptLibrary		
	memScript	Starting from version 4.0.0.
amCbkStoredEvent		
	FieldName	Starting from version 4.0.0.
	Context	Starting from version 4.0.0.
amCbkRule		
	AmountField	Starting from version 4.0.0.
	EvtField	Starting from version 4.0.0.
	Context	Starting from version 4.0.0.
amCbkScript		
	Context	Starting from version 4.0.0.
	memScript	Starting from version 4.0.0.

Other application data to verify

The following application data is neither converted nor verified with 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 from the old-format working database propagated

The structural parameters are defined with AssetCenter Database Administrator.

The structural parameters from the **old-format working database** are propagated to the **4.3.2-format standard gbbase*.*** files during the step where structural changes are propagated.

▸ Step 3 - Propagate structure changes made to the old-format working database [page 48]

Table 8.2. Structural parameters of the old-format working database - list

Database object	Parameter	Available in versions:		
		3.0.1, 3.0.2, 3.1.0, 3.5.1, 3.5.2 and 3.6.0	4.0.0	4.1.0 and higher
Table	Name	Yes	Yes	Yes
Table	Description	Yes	Yes	Yes
Table	Hierarchy	No	No	Yes
Table	Can have features	No	Yes	Yes
Table	String	Yes	Yes	Yes
Table	Validity	Yes	Yes	Yes
Table	Relevance	No	Yes	Yes
Field or link	Name	Yes	Yes	Yes
Field or link	Description	Yes	Yes	Yes
Field	Size	Yes	Yes	Yes
Field	Updated	No	Yes	Yes
Field or link	History	Yes	Yes	Yes
Field or link	Read only	Yes	Yes	Yes
Field or link	Mandatory	Yes	Yes	Yes
Field or link	Irrelevance	No	Yes	Yes
Field	Formatting	Yes	Yes	Yes
Field or link	Default value	Yes	Yes	Yes
Field or link	Reapply default value in case of duplication.	Yes	Yes	Yes

Other documentation

This guide only deals with information directly linked to the migration process.

To obtain associated information not covered in this guide, we recommend that you read the following documents:

Table 8.3. Other documentation - list

Document	Information	Format	Location in the AssetCenter installation folder
Differences between the version 3.x and 4.3.2	• List of new features in the version 4.3.2	Printed	**\doc\pdf\diff*.pdf**
		Online	**\doc\pdf\diff*.pdf**
Readme	• Last-minute information.	Text	**readme.txt**
Release Notes	• List of documents provided with AssetCenter • Overview of new functions	Printed	**\doc\pdf\ReleaseNotes*.pdf**
		Online	**\doc\pdf\ReleaseNotes*.pdf**
Installation	• List of AssetCenter programs • Supported operating systems and minimum configuration • Supported DBMSs • Installing AssetCenter	Printed	**\doc\pdf\Installation*.pdf**
		Online	**\doc\chm\install*.chm**
Structure of the database	• List of the database's tables, fields, links and indexes • Agents automatically triggered by AssetCenter	Text file	• **\doc\infos\database.txt** • **\doc\infos\tables.txt**
		Online	**\doc\chm\dbstruct*.chm**

Document	Information	Format	Location in the AssetCenter installation folder
Structural differences between different database versions	• List of tables, fields, links and indexes that have changed.	Text	**\doc\infos\diff*.txt**
		Html	**\doc\infos\diff*.html**
Administering AssetCenter	• AssetCenter Database Administrator • Import	Printed	**\doc\pdf\Administration*.pdf**
		Online	**\doc\chm\admin*.chm**
Advanced use	• Data export	Printed	**\doc\pdf\AdvancedUse*.pdf**
		Online	**\doc\chm\advanced*.chm**

For more information about XML, consult the Web site: http://www.w3.org/XML/.

Index

; (special character), 131
!-- (migration.xml), 131
!--- (migration.xml), 131
/ (character), 42
^ (character), 40
' (special character), 131
" (special character), 131
& (special character), 131
< (special character), 131
> (special character), 131
|| (special character), 131

A

Access restrictions, 100
acmig.dtd, 139 , 124
Adjustments, 62
Administration, 149
Advanced use, 149
Analyze only, 68
Application data
  (See Also Conversion)
  (See Also Structure - propagating changes)
  Application data to be converted without AssetCenter Script Analyzer, 146
  Application data to be manually converted - list, 142
  Conversion file, 138
  Conversion rules, 60
  Correction - speeding up, 77
  Definition, 119
  Export (See Application data - exporting)
    Process, 70
    Rules, 71
  Restoring, 93 , 85
    Process, with AssetCenter Database Administrator, 85
    Process, witn AssetCenter Script Analyzer, 86
  Verification and correction, 74
    Process, 74
  Verifying the restored data, 87
AQL queries, 143
AssetCenter clients, 115
AssetCenter Database Administrator
  Application data - exporting, 70
  Application data - restoring, 85
  Database - copy, 53
  Database integrity - verification, 67 , 35
  Structural parameters - propagation, 147
  Structure - propagating changes (See Structure - propagating changes)
AssetCenter Export, 146
AssetCenter programs - updating, 111-115
  Process, 112
AssetCenter screens, 98
AssetCenter Script Analyzer
  .xml files, 79
  Application data - restoring, 86

Application data - verifying and correcting, 74
Checked application data, 143
Menus, 78
Scripts - problems, 82
Unchecked application data, 146
AssetCenter Server, 112
AssetCenter Web, 146 , 108
Assets, 139 , 103 , 61
Available guides, 148

B

Basic scripts, 143 , 82
Brands, 102 , 102 , 66 , 66
Budgets, 107 , 105 , 104 , 47
Buttons, 98

C

Caches, 112
Calculated strings, 143
Catalog, 106
Categories, 47
Chargeback, 107
Checking validity of records (option), 68 , 37 , 36
config (folder), 100 , 96
Connections, 115
Connect-It, 146 , 109 , 33
Connect-It scenarios, 109
Consumptions, 62
continueonerror (migration.xml), 124
Conversion, 92
Conflicts, 65
Conversion machine, 32
Customization - finalization, 98
Definition, 118
Final conversion, 91
Manual conversion - reasons for, 22
Process, 55
What's new, 26
Conversion file
Adapting, 121 , 54
Application data, 138
Assets, 139
Attributes
continueonerror, 124
encoding, 124
engine, 125
SYSTEM, 124
Conversion rules, 122
Definition, 118
Elements
!--, 131
!---, 131
Exception, 130
Field, 129
Mapping, 127
MigrationFile, 124
PostActions, 130
PreActions, 128
Script, 131
StartScript, 125
Translate, 126
Foreign keys, 137
Joins, 136
Mulitiple destination tables, 137
Numeric fields, 137
Special characters
;, 131
', 131
", 131
\, 131
&, 131
<, 131
>, 131
||, 131
SQL statements, 138
Syntax, 123
Verification, 139
What it is used for, 122
Conversion machine
Definition, 118
Preparation, 32
Conversion speed, 33
Conversion tools, 25
Converting the database, 54
Convert the database (menu), 56

Copy database to empty database (menu), 53
Cost categories, 105 , 47
Counters, 38
Countries, 102 , 102
Crystal Reports, 146 , 99 , 99
Currencies, 44
Customizing the database, 98

D

Database
  Blocking and copying, 92
  Conversion, 92
    (See Also Conversion)
  Copy, 52
    AssetCenter Database Administrator, 53
    DBMS tools, 53
    Traditional backup - problems, 52
  Finalization, 93
  Integrity - verification, 93 , 91 , 87 , 67 , 35
  Manual adjustment, 37
  Production phase, 115
  Restriction of certain rights, 69
  Structure, 148
  Structure - differences between versions, 140
  Supported versions, 15
  Unsupported DBMSs, 26
  Validation, 68
Database structure
  Definition, 120
Data - definition, 119
db2udb72.sql, 55
dbbscripts (folder), 75 , 74 , 51
DBMS
  Preparing the server, 34
  Supported DBMSs, 15
  Unsupported versions, 26
DBMS server - preparation, 34
Default values, 58
Definitions
  Application data, 119
  Conversion file, 118
  Conversion machine, 118
  Converting the AssetCenter database, 118
  Data, 119
  Migration, 117
  Structure of the AssetCenter database, 120
  Trigger, 119
  Updating AssetCenter programs, 117
  Working database, 119
diff*.*, 140
diff*.htm, 142
diff*.txt, 140
diff*.xml, 142
Disk space, 16
dtd (file extension), 124

E

Elementary adjustments, 43
encoding (migration.xml), 124
engine (migration.xml), 125
Esimates, 45
Exception (migration.xml), 130
Exit (menu), 78
Export application data (menu), 70
Export scripts, 110
Export - scripts, 110
External programs
  Updating, 108
External tools - integration, 89

F

Features, 135 , 133 , 106 , 101 , 101 , 65
Field (migration.xml), 129
Fields, 143 , 97
  Dividing between several tables, 133
  Fields storing application data to be converted manually, 60
  Fields that store a field name, 143
  Length, 39
  Mandatory fields, 57 , 38
  Numeric fields, 137
  Transferring a feature, 133
Floor plan positions, 57
Force the restorable nature of the file (menu), 79
Foreign keys, 137
Forms, 146 , 98

fromxxx (folder), 118 , 54
Full names, 42
Functional domains, 105 , 43
Functionally valid (option), 88 , 77 , 75
Functional rights, 105

G

gbbase.xml, 120 , 100 , 96
gbbase*.*, 147 , 84 , 75 , 57 , 48 , 38 , 23 , 18
Get-It, 146 , 108
Get-Resources, 146 , 108
Glossary, 117-120

H

Help on fields, 146 , 95
History, 61

I

Import scripts, 146 , 109
Import - scripts, 109
Indexes, 95 , 58
infos (folder), 140
Installation, 148
Installation program, 113
Integrity - verification, 67
Itemized lists, 43

J

Java - installation, 33
Joins, 136

L

License contracts, 106 , 103 , 64 , 44
Line-of-business data - exporting, 71
Links, 101
List all files (menu), 79 , 74
List the problems in the script (menu), 82 , 78 , 75
List unprocessed files (menu), 79 , 74
Locations, 104

M

Mandatory links, 38
Mapping (migration.xml), 127
Message (window), 74
Migration
  AssetCenter - version 4.1.x or earlier
  AssetCenter - version 4.2.x
  Definition, 117
  Final conversion, 91-92
  Final phase, 111-116
  Preparation phase
  Preparatory phase, 29-34
  Process, 17-27
  Simulation, 35-89
migration.xml, 118 , 97 , 94 , 92 , 87 , 80 , 69 , 68 , 52 , 37 , 26 , 26 , 19
  (See Also Conversion file)
MigrationFile (migration.xml), 124
Migration folder (field), 86
Models, 103
modifications.xml, 87 , 80

N

Natures, 103 , 60
New (menu), 78
newdbb.log, 50

O

Open (menu), 78 , 36 , 36
Open existing database (menu), 55 , 50
Open the next file (menu), 78
Open the previous file (menu), 78
Order lines, 47
Output events, 101
Owner (field), 56

P

PostActions (migration.xml), 130
PreActions (migration.xml), 128
Procurement (module), 41 , 27
Product packages, 63 , 46 , 44
Products, 63
Products - packages, 63 , 46 , 44

Products - suppliers, 44
Product suppliers, 44
Propagate the customized structure (menu), 50
Purchase orders, 106
Purchase orders - lines, 47

R

Readme, 148
reference (folder), 73
referencenew (folder), 73
References, 121-149
Release Notes, 148
Repair (option), 37 , 36
Repair database (menu), 36 , 36
Repository, 106
Restorable (option), 76
Restore application data (menu), 86 , 85 , 79
Rights - restrictions, 69
Rollback segments, 34

S

Sample data - exporting, 71
Save (menu), 78
Save as (menu), 78
Script (migration.xml), 131
sdu.log, 56
sdu.xml, 139
sdurest.log, 86 , 85
sduxprt.log, 70
Software, 103
Software installations, 63
Software licenses, 64 , 44
Special characters, 131
SQL statements, 138
SQL - statements, 138
StartScript (migration.xml), 125
Stored procedures, 95
Structural parameters, 57
  Propagation, 147
Structure of the database
  Changes - propagation (See Structure - propagating changes)
Structure - propagating changes, 48
  Conflicts, 52
  Conversion file, 52
  Overview, 49
  Process, 50
Sun Java Runtime - installation, 33
Supported environments, 15-16
Supported operating systems, 15
SYSTEM (migration.xml), 124
System data, 60
  Export, 71

T

Tables
  Conversion order, 59
  Dividing the fields between several tables, 133
  Multiple destination tables, 137
Training, 31
Transaction logs, 34
Translate (migration.xml), 126
Triggers, 95
  Definitions, 119

U

Units, 102 , 102
up_GetCounterVal (stored procedure), 94 , 38
Update, 117
User (field), 56
User profiles, 100
User rights, 100
Use the following mapping to add another currency (mapping.xml), 45

V

Validate the script in its context (menu), 78 , 76
Value (attribute), 58
Views, 146 , 98 , 95

W

Web pages, 146
Wizards, 143
Workflow (module), 41
Working database, 119
Working folder (field), 86 , 86

X

xerces.jar, 52
xml (file extension), 124 , 87 , 86 , 86 , 82 , 79 , 78 , 74 , 70 , 51
XML - editor, 33
XML editor - installation, 33