HP Asset Manager

Software version: 9.32

Migration

Document Release Date: 18 December 2012
Software Release Date: December 2012

Legal Notices

Copyright Notices

© Copyright 1994-2012 Hewlett-Packard Development Company, L.P.

Restricted Rights Legend

Confidential computer software.

Valid license from HP required for possession, use or copying.

Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services.

Nothing herein should be construed as constituting an additional warranty.

HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

For information about third-party and/or open source license agreements, or to view open source code, use one of the following ways:

- In the `ThirdParty` directory of the Asset Manager installation CD-ROM
- In the directories in which the binary files of the third-party and/or open source tools are located after installation of Asset Manager.
- Through the component's url indicated in the Asset Manager **Open Source and Third-Party Software License Agreements** guide

Trademark Notices

- Adobe®, Adobe logo®, Acrobat® and Acrobat Logo® are trademarks of Adobe Systems Incorporated.
- Corel® and Corel logo® are trademarks or registered trademarks of Corel Corporation or Corel Corporation Limited.
- Java is a registered trademark of Oracle and/or its affiliates.
- Microsoft®, Windows®, Windows NT®, Windows® XP, Windows Mobile® and Windows Vista® are U.S. registered trademarks of Microsoft Corporation.
- Oracle® is a registered trademark of Oracle Corporation and/or its affiliates.
- UNIX® is a registered trademark of The Open Group.

Acknowledgements

This product includes software developed by the Apache Software Foundation (http://www.apache.org/ [http://www.apache.org/]), which is Copyright © The Apache Software Foundation. All rights reserved.

This product includes software developed by The OpenLDAP Foundation, which is Copyright ©, The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved. OpenLDAP® is a registered trademark of the OpenLDAP Foundation.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/), which is Copyright © The OpenSSL Project. All rights reserved.

This product includes software developed by the OpenSymphony Group (http://www.opensymphony.com/), which is Copyright © The OpenSymphony Group. All rights reserved.

This product includes code licensed from RSA Data Security.

This product includes software developed by the JDOM Project (http://www.jdom.org/), which is Copyright © Jason Hunter & Brett McLaughlin. All rights reserved.

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

Chapter 2. Prerequisite 17

Warning on using HP Connect-It 17
What does migration entail? 17
What does the conversion entail? 17
What is converted during the migration? 18
Migrating a database whose DBMS is not supported by version 9.32 . 18
Limitations of the Procurement module 18
Complexity of the migration 19

Chapter 3. Step-by-step migration - preparation phase (**production database**) 21

Preliminary analysis 21
Launching the migration project 22
Training the users and support technicians 23
Preparing your conversion computer 24
Preparing the DBMS server 26

Chapter 4. Step-by-step migration - simulation (**simulation database**) 27

1 Step 1 - Verify the integrity of the old-format production database . . 27
2 Step 2 - Manually adjust the old-format production database . . . 29
3 Step 3 - Propagate structure changes made to the old-format production database 33
4 Step 4 - Copy the old-format production database 36
5 Step 5 - Convert the **old-format simulation database** 38
6 Step 6 - Verify the integrity of the **9.32-format simulation database** 45
7 Step 7 - Validate the **9.32-format simulation database**. 46
8 Step 8 - Adapt the integration with external tools 47

Chapter 5. Step-by-step migration - final conversion (**migration database**) 49

9 Step 9 - Block and copy the old-format production database 49
10 Step 10 - Finalize the **9.32-format migration database** 50
11 Step 11 - Upgrade the external software components that access the Asset Manager database 57

Chapter 6. Step-by-step migration - final phase . . . 59

12 Step 12 - Upgrade the Asset Manager programs 59
13 Step 13 - Put the **9.32 format migration database** into production 61
14 Step 14 - Uninstall programs no longer used 61

Chapter 7. Glossary 63

Migration 63
Updating Asset Manager programs 63

Converting the old-format production database 64
Conversion file 64
Conversion machine 64
Production database 64
Trigger 65
Data 65
Application data 65
Database structure 65

Chapter 8. References 67

Adapting the `migration.xml` conversion file 67
Structural modifications to the standard database compared to previous versions 84
Structural parameters from the **old-format production database** propagated 86
Other documentation 87

Index 89

List of Figures

4.1. Propagating structural changes - process 34

List of Tables

1. Operations to perform depending on the previous version number . . 12
4.1. Fields that must not contain the ^ character - list 31
8.1. Structural parameters of the **old-format production database** - list 86
8.2. Other documentation - list 87

Introduction

The operations to be performed to upgrade from a previous version of Asset Manager to version 9.32 depend on the number of your previous version.

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

Important:

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

The more complex cases are referred 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

Number of the version to upgrade	Operations to perform	Relevant documentation
Version 5.10 or later	In standard situations, a **simple upgrade** will suffice.	Refer to **Installation and upgrade** guide, chapter **Upgrading a previous version**, section **Upgrading Asset Manager version 4.2.x, 4.3.x, 4.4.x or 5.0x - Overview**.
	If the simple upgrade fails, you will need to perform a **simplified migration**.	Follow the instructions in this guide.
Versions earlier than 5.10	Versions earlier than 5.10 cannot be directly migrated to 9.32 or later versions. You must migrate to version 5.10~9.31, and then migrate to the latest version of Asset Manager.	Refer to the **Migration** guide shipped with Asset Manager version 5.10 to 9.31.

Why migrate?

The newer version of Asset Manager is improved with new structural modifications:

- The standard database structure (tables, fields, links, indexes) has been vastly modified.
- New functions have been added.
- Schema changes have been introduced by the content packages such as Cloud Billing and SAM.

All these changes have made it necessary to methodically migrate your earlier version of Asset Manager to the 9.32 version.

What does migration involve?

Migration involves performing the following tasks:

- Converting the old-format production database to the 9.32 format (structure and content).
- Upgrading the Asset Manager programs to the version 9.32.

Who is migration intended for?

This migration is performed by the engineers in charge of:

- Administering the Asset Manager database.
- Installing Asset Manager.
- Deploying Asset Manager.

Required competencies

Migration is a complex process that requires:

- A thorough understanding of the earlier versions of Asset Manager and of version 9.32 (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 Asset Manager 9.32 guides:

- **Installation and upgrade**
- **Release Notes**
- **Migrating customized compact SI from AC 4.4.x to AM 5.1x** (located in your <Asset Manager 9.32 installation folder>\doc\white_papers\Software Asset Management)

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 Prerequisite

This chapter provides an overview of the migration process.

This process will differ depending on which version of Asset Manager 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 (production 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.

1 Supported environments

Operating systems and DBMSs

This migration works with all operating systems and DBMSs supported by Asset Manager.

To learn which operating systems and DBMSs are supported, refer to the Support Matrix on the Web site. www.hp.com/go/hpsoftwaresupport.

2 Prerequisite

Warning on using HP Connect-It

You must not use HP Connect-It to convert the **old-format production database**.

What does migration entail?

This migration is a set of operations required to convert an earlier version of Asset Manager to the version 9.32:

- Converting the **old-format production database** (structure and contents) in order to make it compatible with the 9.32 version of Asset Manager.
- Updating the Asset Manager programs to the version 9.32 on all administration and user machines.

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

What does the conversion entail?

Converting a database entails:

- Making the structure of the current database conform to that of the 9.32 version of Asset Manager.
- 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 during the migration?

- 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.

Migrating a database whose DBMS is not supported by version 9.32

If the DBMS of the old-format production database is not supported by version 9.32:

1. Transfer the old-format production database to a DBMS that is supported by Asset Manager 9.32.

   To learn how to do this, refer to the **Administration** guide, chapter **Creating, modifying and deleting an Asset Manager 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 production 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 production 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 Asset Manager, however.

The complexity of the conversion depends on the degree of customizations made to the old-format production database.

3 Step-by-step migration - preparation phase (production 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 Asset Manager 9.32.

   ▸ Other documentation [page 87]
3. Determine what impact these modifications (new functions, modifications of functions, etc.) will have on your use of Asset Manager.
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 9.32 of Asset Manager, the migration process needs to involve those people in charge of:

- Nomenclature
- Deploying the functional modules:
  - Procurement
  - Contracts
  - Financing
  - Helpdesk
- Inventory
- Customizing the database.
- Creating, reports, queries, workflow schemes, actions, etc.
- Integrating Asset Manager 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 Asset Manager 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 Asset Manager and stabilizing this.
2 Exploring the new functionality in Asset Manager 9.32.

This will ensure a smoother transition.

Tip:

Do not hesitate to call on HP 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 production database, you might also want to think about training those people who use and support the use of Asset Manager.

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 Asset Manager need to be trained before you can put the **9.32-format production database** into production.

Tip:

Do not hesitate to call on HP 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 production database, you must prepare a computer adapted for this conversion.

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

Installing the Asset Manager version corresponding to the old-format production database

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

- **Production database**
- **Simulation database**
- **Migration database**

At the least, you must install the basic module.

Verify that you have access to the old-format production database

You need to do this in order to:

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

Installing Asset Manager 9.32

You need to install at least the following components:

- Asset Manager client
- Asset Manager Application Designer
- Documentation
- Migration
- Datakit
- Asset Manager Export Tool
- HP AutoPass License Management Tool

  All the License Keys that apply to the database must be installed on this instance of HP AutoPass License Management Tool.

  ▸ The **Administration** guide, chapter **Installing License Keys**.

Tip:

HP AutoPass License Management Tool is automaticaly installed with any Asset Manager 9.32 component that you install.

Installing License Keys is a manual process, though.

Note:

If your **old-format production database** is multilingual (▸ **Administration** guide, chapter **Creating, modifying and deleting an Asset Manager database**, section **Modifying Asset Manager client languages**), and you have customized certain multilingual items and you want to automatically convert the multilingual items (▸ Propagate the structural changes. [page 35]), Asset Manager 9.32 must be available in the additional languages and you must install Asset Manager in these languages on the computer used for the conversion.

All multilingual elements will be propagated, except for contextual help on fields and links (▸ Help on fields [page 51]).

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 Java Runtime environment (the version provided with Asset Manager 9.32)

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

Increase Java's heap size to avoid memory issues:

1. Locate the `amdba.ini` file: ▸ **Installation and upgrade** guide, chapter **.ini and .cfg files**, section **Available .ini and .cfg files**.
2. Open the file in a text editor.
3. In the **[Option]** section, add or modify the **/Advanced/SduJavaCmd** parameter and set its value to **java -Xmx500M** : **/Advanced/SduJavaCmd=java -Xmx500M**.
4. Save `amdba.ini`.

Factors affecting the conversion rate

- DBMS performances
- Throughput between the Asset Manager Application Designer machine and the machine of the old-format databases.
- Performances of the machines where Asset Manager Application Designer and the old-format databases are installed (but only minimally).

Tip:

If you have a large volume of data in the old-format production database, you must position the computers where Asset Manager Application Designer 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.

Rollback segments

Note:

Rollback segments is the terminology used by Oracle.

Its equivalent in Microsoft SQL Server is **transaction logs**.

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

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

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

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

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

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

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

It is this **9.32-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 production database

1

Important:

Make a backup of the old-format production database.

2 Perform an optional initial verification with the old-version Asset Manager Application Designer:

Warning:

This check is optional.

It may take more than a day to check the **Check validity of records** option for tables that have a **Validity** script and contain many records.

For such tables, the validity script will be executed for each record in the table.

In some instances, the check may never get through.

1 Launch the old-version Asset Manager Application Designer.
2 Connect to the **old-format production database** (**File/ Open** menu, **Open existing database** option).
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 Only select the **Check 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.

3

Warning:

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

Perform a second verification with 9.32-format Asset Manager Application Designer:

1 Launch Asset Manager Application Designer 9.32.
2 Connect to the **old-format production database** (**File/ Open** menu, **Open existing database** option).

Note:

It is fully possible to connect to the previous format database using Asset Manager Application Designer 9.32.

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 available options, make sure that the **Check validity of record** option is NOT selected.
7 Select the **Repair** option.
8 Click **Start**.
9 Consult the messages of the execution window.
10 Consult the log file if necessary.

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 production database

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

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

Most of the constraints to respect in the **old-format production 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 production 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
- Oracle Database Server: TN12516652
- DB2 UDB: TN1029175140 (for Asset Manager 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 production 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.

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 Asset Manager 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 9.32 gbbase*.*` database-description files.

Fields and links must have an explicit association (described in the `migration.xml` conversion file) or an implicit association (automatically deduced when fields or links share the same SQL name).

The `migration.xml` conversion files installed by default with Asset Manager 9.32 are intended to work properly when the format of the **old-format production database** and the database-description files, `standard 9.32 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 production database.
- If you added the mandatory nature to certain fields or links of the `standard 9.32 gbbase*.*` database-description files.

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

You must make sure that the fields and links that are declared mandatory in the `customized 9.32 gbbase*.*` database-description files are populated in the old-format production 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 production database are used to populate other fields in the 9.32-format production 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 9.32 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 production 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 production 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

SQL name of the table	SQL name of the field
amDateAlarm	Code
amDeprScheme	Code
amEscSchLevel	Code
amFloorPlan	Code
amFuncDomain	SQLName
amFuncDomain	Name
amReservation	ItemNo
amLocation	BarCode
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.).

Warning:

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

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

Warning:

To perform this operation, the `standard 9.32 gbbase*.*` database-description files that you use here must be the standard file installed with Asset Manager 9.32. You cannot use these files if any modifications were made

This operation:

- Concerns the users who modified the standard structure of the old-format production database (addition or modification of fields, indexes and tables) and want to keep those changes in the 9.32-format production database.
- Aims to propagate the structural modifications in the `standard 9.32 gbbase*.*` files.

  Tip:

  The `standard 9.32 gbbase*.*` files obtained will be used to structure the **9.32-format database** during the conversion.

- Uses a tool dedicated to this operation, which is available in Asset Manager Application Designer.

Warning:

Only the structural changes made to the old-format production database using Asset Manager Application Designer will be accounted for.

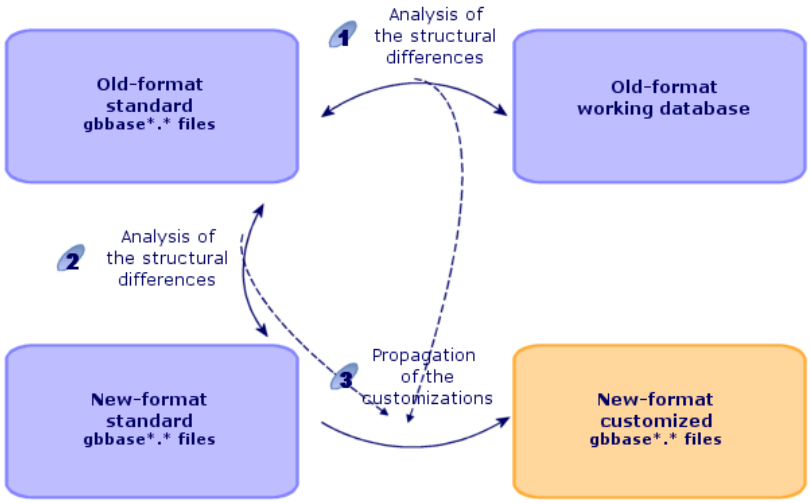
You must manually delete all structural changes made by any other means in the old-format production database.

List of propagated structural parameters: ▸ Structural parameters from the old-format production database propagated [page 86].

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 production database and the `standard old-format gbbase*.*` files.

2: The tool determines the differences between the `standard old-format gbbase*.*` files and the `standard 9.32 gbbase*.*` files.

3: The tool copies and modifies the `standard 9.32 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 9.32 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 production database** - before definitively propagating the structural changes - you must modify the SQL names of tables, fields and indexes that appear in the version 9.32.

  Otherwise they will conflict with the standard field of the version 9.32 with the same name.

Propagate the structural changes.

1. Launch Asset Manager Application Designer 9.32.
2. Connect to the old-format production database with the **Admin** login (**File/ Open/ Open existing database** menu).
3. Select the **Migration/ Propagate the customized structure** menu.

   Note:

   If the **old-format production database** is multilingual (▸ **Administration** guide, chapter **Creating, modifying and deleting an Asset Manager database**, section **Modifying Asset Manager client languages**), one of the pages in the wizard offers to propagate the customizations made for the additional languages of the **old-format production database**. This requires Asset Manager version 9.32 to be available in each of the additional language versions, and that you install Asset Manager in these languages on the computer used for the conversion.

   All multilingual elements will be propagated, except for contextual help on fields and links (▸ Help on fields [page 51]).
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.
6. If the messages tell you so, modify the structure of the old-format production database. Then perform the migration starting from the step Step 4 - Copy the old-format production database [page 36].

   This must be repeated until you obtain a good `customized 9.32 gbbase*.*` files without any problem messages.
7. Certain scripts might not be propagated to the `standard 9.32 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 9.32 gbbase*.*` files.

Warning:

However, you will need to modify the `customized 9.32 gbbase*.*` files again when you execute the step Step 5 - Convert the old-format simulation database [page 38].

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.

Step 4 - Copy the old-format production database

Problems that can occur during a traditional backup

If you make a backup of the old-format production database using DBMS tools, the backup of the old-format production database will be identical to the original for everything concerning additions, modifications or deletions of the following events using tools other than Asset Manager Application Designer:

- 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 production 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 production database in an empty database using Asset Manager Application Designer.

Note:

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

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

   The backup is identical to the original old-format production database.
2. Delete all the modifications made to:
   - Indexes
   - Triggers
   - Stored procedures
   - Views
3. Create an Asset Manager connection to **old-format simulation database**.

Solution 2: Copy the old-format production database into an empty database using Asset Manager Application Designer

1. Create an empty, old-format Asset Manager database.
2. Create an Asset Manager connection to this empty database.
3. Open the **old-format production database** in Asset Manager Application Designer.
4. Copy the **old-format production 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 production database in an empty database using Asset Manager Application Designer, refer to the **Administration** guide, chapter **Using a test database**, section **Copying your production database**.

Step 5 - Convert the **old-format simulation database**

> Warning:
>
> The conversion tools must not be used to modify the structure of the **9.32-format production database** (adding, deleting or modifying tables, fields, indexes, stored procedures, triggers, screens, etc.).
>
> Such modifications must be planned after the migration.

Adapt the `migration.xml` conversion file

> Warning:
>
> This operation must be carried out by a HP certified technician for the migration.
>
> HP declines all responsibility if this condition is not strictly adhered to.

Asset Manager 9.32 is installed with conversion files by default (1 file per earlier version of Asset Manager 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\HP\Asset Manager 9.32 xx\migration\fromxxx` folder, where `xxx` is the number of the earlier version.

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

If you have particular needs (fields performing 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 67].

Constraints caused by modifying the data in the **old-format production database**: ▸ Step 2 - Manually adjust the old-format production database [page 29].

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.

Prerequisite if the **old-format production database** is inferior to version 5.20 of Asset Manager and uses Oracle

Starting with version 5.20, Asset Manager database based on Oracle employs **CLOB/BLOB** data types for fields that previously used **LONG** and **LONGRAW** respectively. Thus it is necessary to identify these fields and change their data types before the database conversion can proceed.

To change the data types for the database:

1 Launch Asset Manager Application Designer version 9.32.

2 Connect to the **old-format simulation database** using the **Admin** login (**File/ Open/ Open an existing database**).

Important:

In the connection detail of Asset Manager:

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

3 Select **Action/ Templates/ Select folder...** from the menu bar.

4 Select the `<Asset Manager 9.32 installation folder>\doc\infos` folder and click **OK**.

5 Select **Action/ Templates/ Refresh list** from the menu bar.

This adds a new option called **ORACLE batch for BLOB migration** to the **Templates** menu, based on the `migratelob.tpl` template file.

6 Select **Action/ Templates/ ORACLE batch for BLOB migration** from the menu bar.

This generates an Oracle SQL+ batch file called `migratelob.sql` by default, which contains instructions to convert **LONG** and **LONGRAW** fields to **CLOB** and **BLOB** respectively.

7 Use a database utility such as ORACLE SQL+ Prompt to run the `migratelob.sql` batch file. Example:

```
SQL> @C:\Users\encornet\AppData\Local\Temp\migratelob.sql
```

This changes fields to the new data types, after which the standard database upgrade can proceed.

Important:

If you have developed solutions that directly access the Asset Manager database (through an ODBC connection), you will need to update the integration where your solution accessed **LONG** and **LONGRAW** data types, after the **copy of the old-format production database** has been converted to version 9.32.

Convert the **old-format simulation database**

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

1. Launch Asset Manager Application Designer version 9.32.
2. Connect to the **old-format simulation database** with the **Admin** login (**File/ Open/ Open existing database** menu).

   Important:

   In the connection detail of Asset Manager:
   - 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**).
3. Select **Migration/ Convert the database**.
4. 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).
5. Consult the messages of the `sdu.log` log file.

Warning:

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

1 Correct the source of the problem.

2 Restart the conversion from step Step 4 - Copy the old-format production database [page 36].

Information about the conversion

Here are some rules that are used during the conversion.

Tip:

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

Rules used for all source versions of the **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 9.32 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 production database.

Default values of fields

The default values defined in the structure of the production 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 9.32?	Are there modifications to fields, links or indexes between the old version and version 9.32?	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.

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 **Utilities/ Configure list** from the contextual menu.

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

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 Asset Manager.

Note:

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

An example of their usefulness: When exporting data from Asset Manager to be modified outside of Asset Manager then reimported in Asset Manager. 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 **9.32-format simulation database**

1 Important:

Make a backup of the old-format production database.

2 Launch Asset Manager Application Designer 9.32.

3 Connect to the **9.32-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 **Check 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.

If problems are displayed by the program, follow these steps:
1 Modify the data in the old-format production database.
2 Start again from step Step 4 - Copy the old-format production database [page 36].

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

Step 7 - Validate the **9.32-format simulation database**.

Browse the **9.32-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 **9.32-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 sensitive 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 38].

Or:

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

2 Start again from step Step 4 - Copy the old-format production database [page 36].

Step 8 - Adapt the integration with external tools

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

Potentially concerned applications: ▸ sections:

- Asset Manager Web [page 57]
- HP Connect-It scenarios [page 57]

You only implement the new integration mode in these applications after the step Step 10 - Finalize the 9.32-format migration database [page 50].

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

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

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

At this stage, you have:

- A set of `customized 9.32 gbbase*.*` files.
  - Step 3 - Propagate structure changes made to the old-format production database [page 33]
- A `migration.xml` conversion file that was tested on the **simulation database**.
- Manually converted application data that was tested in the **9.32-format simulation database**.

This chapter explains step-by-step which operations to perform to convert the **9.32-format production database**.

Step 9 - Block and copy the old-format production database

Blocking the old-format production 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 production database.
2. Shut down the:
   - Asset Manager Automated Process Manager
   - Asset Manager APIs

- External programs that access the old-format production database.

3 Block access to the old-format production database.

4 Make a backup of the old-format production database as described in the section Step 4 - Copy the old-format production database [page 36].

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

You need to minimize the time the old-format production 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 10 - Finalize the **9.32-format migration database**

You will need to make alterations to the **9.32-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 **9.32-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 **9.32-format migration database**.

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

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 **9.32-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 production database.

Before converting the old-format production 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
- Oracle Database Server: TN12516652
- DB2 UDB: TN1029175140 (for Asset Manager versions 3.x)

Triggers, indexes, stored procedures and views

Before the conversion, you put the old-format production 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 Asset Manager 9.32.
   2. Connect to the **9.32-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 database with the DBMS of your choice.

     To learn how to create an empty database, refer to the **Administration** guide, chapter **Creating, modifying and deleting an Asset Manager database**, section **Creating an empty shell with the DBMS**.
   2 Start the earlier version of Asset Manager.
   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 9.32.

1 Start Asset Manager Application Designer.
2 Select the **File/ Open** menu.
3 Select the **Open database description file - create new database** option.
4 Select the `standard 9.32 gbbase.xml` file, located in the `config` sub-folder of the Asset Manager 9.32 installation folder.
5 Start the database creation wizard (**Action/ Create database** menu).
6 Populate the pages of the wizard as follows (navigate through the wizard pages using the **Next** and **Previous** buttons):

   Generate SQL script / Create database page:

Fields	Value
Database	Select the connection to the **9.32-format migration database**.
Creation	Import line-of-business data
Use avanced creation options	Select this option.

Creation parameters page:

Fields	Value
Password	Administrator password. Note: The Asset Manager database adminstrator is the record in the **Employees and departments** (amEmplDept) table for which the **Name** (Name) field is set to **Admin**. The database connection login in stored in the **User name** (UserLogin) field. The administration name is **Admin**.

Create system data page:

Fields	Value
Use time zones	Do not select this option
Use help on fields	Select this option.

Data to import page:

Fields	Value
Available data	Do not select any data.
Stop import if error	Do not select this option
Log file	Do not populate this field.

7 Execute the options defined using the wizard (**Finish** button).

8 Examine the messages in the **Database creation** page and then click **OK** to close.

Reapplying customizations to help on fields

By updating the help on fields of version 9.32, you are overwriting the customizations that you have already made.

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.

User rights, access restrictions and functional rights

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

Add the new tables, fields and links to the existing rights and restrictions and create new rights and restrictions if necessary.

Transferring certain features to fields

Asset Manager 9.32 enables you to access new fields, whether they come from the standard 9.32-format database structure or customizations of your own.

You may want to use one of these new fields instead of a feature used in the **old-format production 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.

Procedure

For information on this procedure, refer to the **Administration** guide.

Views

A screen was created for all shared views.

You can delete obsolete views.

SAP 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 become no longer valid.

Reusing previous reports

1 Launch Asset Manager.
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 SAP Crystal Reports according to its message.

5 If you want to import the new, standard reports provided with Asset Manager 9.32:

   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 Asset Manager.
2 Display the list of reports (**Tools/ Reporting/ Reports** menu).
3 Delete all the previous reports.

Importing the standard reports provided with Asset Manager 9.32

To import the **sample data** reports in the **9.32-format migration database**:

1 Start Asset Manager Application Designer.
2 Select the **File/ Open** menu.
3 Select the **Open database description file - create new database** option.
4 Select the `standard 9.32 gbbase.xml` file, located in the `config` sub-folder of the Asset Manager 9.32 installation folder.
5 Start the database creation wizard (**Action/ Create database**).

6 Populate the pages of the wizard as follows (navigate through the wizard pages using the **Next** and **Previous** buttons):

Generate SQL script / Create database page:

Fields	Value
Database	Select the connection to the database into which you wish to import the reports.
Creation	Import line-of-business data.
Use avanced creation options	Do not select this option

Creation parameters page:

Fields	Value
Password	Enter the administrator's password. Note: The Asset Manager database adminstrator is the record in the **Employees and departments** (amEmplDept) table for which the **Name** (Name) field is set to **Admin**. The database connection login in stored in the **User name** (UserLogin) field. The administration name is **Admin**. The password is stored in the **Password** field (LoginPassword).

Data to import page:

Fields	Value
Available data	Select the option **Crystal Reports**.
Stop import if error	Select this option for the import to stop if a problem is encountered.
Log file	Full name of the file to which all import operations, including errors and warnings, are logged.

7 Execute the options defined using the wizard (**Finish** button).

Step 11 - Upgrade the external software components that access the Asset Manager database

Asset Manager Web

You must uninstall your old version of Asset Manager Web and install version 9.32.

Asset Manager Web has been completely redesigned since 5.0.0. Web client screens are now the same as those found in the Windows client (with the exception of a few administration screens).

As for the Windows client, Asset Manager Application Designer is used to customize Web client screens.

All customizations that were made in the previous version of Asset Manager Web will be lost.

HP Connect-It scenarios

To access the **9.32-format migration database** using HP Connect-It, you must use the version of HP Connect-It supported by Asset Manager 9.32.

If you use standard HP 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 HP Connect-It.
3. Open each scenario one by one in HP Connect-It.
4. For each scenario:
    1. Examine the possible warning messages displayed by HP 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.

6 Step-by-step migration - final phase

This chapter explains step-by-step which operations to perform to get your **9.32-format migration database** up and running.

Step 12 - Upgrade the Asset Manager programs

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

You must also make sure that the version of the programs that interact with Asset Manager are still compatible with Asset Manager 9.32. If necessary, upgrade these programs as well.

To obtain a list of Asset Manager programs and other programs that interface with Asset Manager, refer to the **Installation and Upgrade** guide, chapter **Components of Asset Manager**.

To learn which program versions are compatible with Asset Manager 9.32, consult the Support Matrix at: www.hp.com/go/hpsoftwaresupport.

Install Asset Manager Automated Process Manager on an administration machine

Asset Manager Automated Process Manager carries out a number of automatic tasks on the Asset Manager database. If it is not launched, Asset Manager cannot function correctly.

You must therefore:

1. Install Asset Manager Automated Process Manager on a client machine.
2. Properly configure Asset Manager Automated Process Manager.
3. Execute Asset Manager Automated Process Manager permanently.

To learn more about how Asset Manager Automated Process Manager works, refer to the **Administration** guide, chapter **Asset Manager Automated Process Manager**.

Delete the Asset Manager caches of the **9.32-format migration database**

If you use a cache with the connection to your **9.32-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 **Asset Manager performances**.

Upgrade Asset Manager programs

To upgrade the programs:

1. Uninstall the earlier version of Asset Manager.

   Tip:

   If you are installing Asset Manager 9.32 on a conversion machine, be sure to conserve your previous version of Asset Manager for the time being.

   For information on the uninstallation procedure (safeguards, steps to follow, and ways to remove Asset Manager), refer to the **Installation and upgrade** guide corresponding to the version of Asset Manager to be removed.
2. Install Asset Manager 9.32.

   For information on the installation procedure (safeguards, steps to follow, and ways to install Asset Manager), refer to the Asset Manager 9.32 **Installation and upgrade** guide.

Verify that Asset Manager can be launched without problems

If you are having problems launching Asset Manager 9.32, contact user support.

Remove the old connections to databases and create new ones

The objective is to have the users connect to the **9.32-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 Asset Manager cache for your connections if you consider this will be useful.

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

Refer to the **Tailoring** guide, part 1 **Customizing client computers**, chapter **Customizing a client workstation**.

13 Step 13 - Put the **9.32 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 **9.32-format migration database**.
- Upgraded the Asset Manager programs on all user and administration machines.

Now you must perform the following tasks:

1. Put Asset Manager Automated Process Manager into production on the finalized **9.32-format migration database**.
2. Relaunch the external programs that access the **9.32-format migration database**.
3. Inform users that they can use the database.

14 Step 14 - 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 87]).

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

- The version of Asset Manager corresponding to the **old 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:

- HP Connect-It
- XML file editor
- 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 Asset Manager 9.32 and the production database.

7 Glossary

Migration

The migration is a set of operations required to convert an earlier version of Asset Manager to the version 9.32:

Migration includes:

- Converting the old-format production database (structure and contents) in order to make it compatible with the 9.32 version of Asset Manager.
- Updating the Asset Manager programs to the 9.32 version on all administration and user machines.

Updating Asset Manager programs

One of the operations required by the Asset Manager migration.

Updating the programs involves reinstalling all the Asset Manager programs on all administration and user machines so that they are a version 9.32.

Do not confuse with ...

▸ Converting the old-format production database [page 64]

Converting the old-format production database

One of the operations required by the Asset Manager migration.

Converting the old-format production database involves modifying its structure and contents in order to make it compatible with the 9.32 version of Asset Manager.

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

Do not confuse with ...

▸ Updating Asset Manager programs [page 63]

Conversion file

A conversion file is a file that describes which data to transform during the conversion of the old-format production database and what transformations to perform.

The conversion files are named `migration.xml`.

They are generally located in the `C:\Program Files\HP\Asset Manager 9.32 xx\migration\fromxxx` folder.

Asset Manager is installed with conversion files by default (1 file per version of Asset Manager 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 production database to the 9.32 format.

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

Production database

The production database is the Asset Manager 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 Asset Manager 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 production 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 9.32.

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 Asset Manager Application Designer.

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

8 References

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 Asset Manager, as well as the 9.32 version.
>
> Thus, the adaptation of the conversion file can only be done by a HP-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 HP.

> **Tip:**
>
> Keep in mind that HP 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 64].

To learn when a conversion file needs to be adapted: ▸ Adapt the migration.xml conversion file [page 38].

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 transferred to a direct field or a table in the **9.32-format database**.

Important:

If you wish to convert into fields any other features that are not covered in the standard mappings, do not create new mappings. Refer to section Transferring certain features to fields [page 54].

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 9.32 version of Asset Manager (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]" havin
g="[Q]" autofill="[L]">
<PreActions engine='[T]'>
[U]
</PreActions>
<Field sqlname="[E]" value="[F]"  translate="[X] feature="[Y]" featuretabl
e="[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.

Asset Manager 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 four 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:

- `MSSQL`
- `Oracle`
- `DB/2`

You must respect the case.

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).

Warning:

The AQL language of Asset Manager 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.lPo
rtfolioItemId FROM amAssetOld a, amPortfolio p WHERE a.lParentId=0 AND p.l
AstId=a.lAstId)
GO
DELETE FROM amItemListVal WHERE lItemListId=(SELECT lItemListId FROM amIte
mizedList 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 WHER
E oldsqlname = amDocument.DocObjTable ) WHERE amDocument.DocObjTable IN( S
ELECT 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 9.32 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, ..., [SQ
L name of table n] aliasn"
```

to attribute

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

where attribute

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

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

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

For example, in the following association:

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

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

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

orderby attribute

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

groupby attribute

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

having attribute

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

autofill attribute

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

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

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

PreActions element

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

A PreActions element is useful when you:

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

- Create a feature.

  Important:

  If you wish to convert into fields any other features that are not covered in the standard mappings, do not create new mappings. Refer to section Transferring certain features to fields [page 54].

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 production 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 **9.32-format database**.

Important:

If you wish to convert into fields any other features that are not covered in the standard mappings, do not create new mappings. Refer to section Transferring certain features to fields [page 54].

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 **9.32-format database**.

Important:

If you wish to convert into fields any other features that are not covered in the standard mappings, do not create new mappings. Refer to section Transferring certain features to fields [page 54].

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 is 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. To convert the [Y] feature values to a field, the amAsset table must be declared at the level of the from attribute. And the amFVAsset table is declared 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.

> Important:
>
> If you wish to convert into fields any other features that are not covered in the standard mappings, do not create new mappings. Refer to section Transferring certain features to fields [page 54].

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 production 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 production 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="lAssetRent-Id"`	`\"`	`value="'\"'"`
`'`	Delimits the SQL text string inside the value of an attribute.	`value="soft.Pub-lish-er+'/'+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="'\\'"`
`||`	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 Asset Manager 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 Asset Manager 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

Asset Manager 9.32 includes new fields.

In certain cases, these new fields are used instead of features used in the old-format production database.

Important:

The information in this section is useful to understand the syntax of the existing mappings.

On the other hand, if you do not wish to convert other features to fields, do not create new mappings, but refer to section Transferring certain features to fields [page 54].

Syntax

```
<Mapping to="[SQL name of the destination table]" from="[SQL name of the s
ource table that stores the feature values]">
<Field sqlname="[SQL name of the destination field]" value="[SQL name of t
he field that stores the feature values]"  feature="[SQL name of the sourc
e feature]" featuretable="[SQL name of the table that stores the feature v
alues]"/>
</Mapping>
```

Aliases are used 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" feature
table="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.

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

  Otherwise, the deletion is performed manually after the conversion.

To convert multiple features, we have chosen to use the `<Script>` element, as shown in the following example:

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

Potential problem with link-type features

When transferring the values of the link-type features to a link in the 9.32 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, a query must be executed to identify the records whose links (the ones that replaced the features) are not populated.

Converting a field that stores application data 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 Asset Manager Application Designer 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" where=
"s.lSoftId = soft.lSoftId">
<Field sqlname="lCatProductId" value="s.lProdSoftId"/>
<Field sqlname="InternalRef"   value="soft.Publisher+'/'+soft.Name+'/'+sof
t.VersionLevel"/>
<Field sqlname="FullName"      value="'/'+soft.Publisher+':'+soft.Name+':'
+soft.VersionLevel+'/'"/>
<Field sqlname="dtLastModif"   value="s.dtLastModif"/>
</Mapping>
```

Note:

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

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

Populating foreign keys

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

Example

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

Dividing source tables between two or more destination tables

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

Structural modifications to the standard database compared to previous versions

Asset Manager 9.32 is installed with files (`diff*.*`) that describe the differences in the database structures between:

- Version 9.32.
- A given previous version.

Warning:

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

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

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

They are generally located in the `C:\Program Files\HP\Asset Manager 9.32 xx\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 Asset Manager>.*`

Tip:

You can find the version number by launching the old version of Asset Manager and opening the **Help/About Asset Manager** 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 9.32-format standard database>
```

  - Deletion of table:

```
<SQL name, or if that doesn't exist, technical name in the old-forma
t 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-forma
t standard database> (<SQL name in the 9.32-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-forma
t standard database>
```

  - Object modified:

```
<SQL name, or if that doesn't exist, technical name in the old-forma
t standard database> (<SQL name in the 9.32-format standard database
>)
```

  - Object added:

```
<SQL name in the 9.32-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.

Structural parameters from the **old-format production database** propagated

The structural parameters are defined with Asset Manager Application Designer.

The structural parameters from the **old-format production database** are propagated to the `9.32-format standard gbbase*.*` files during the step where structural changes are propagated.

▸ Step 3 - Propagate structure changes made to the old-format production database [page 33]

Table 8.1. Structural parameters of the old-format production 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

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	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 8.2. Other documentation - list

Document	Information	Format	Location in the Asset Manager installation folder
Readme	◆ Last-minute information.	Text	`readme.txt`
Release Notes	■ List of documents provided with Asset Manager ■ Overview of new functions	Printed	`\doc\pdf\ReleaseNotes*.pdf`
		Online	`\doc\pdf\ReleaseNotes*.pdf`

Document	Information	Format	Location in the Asset Manager installation folder
Installation	■ List of Asset Manager programs ■ Supported operating systems and minimum configuration ■ Supported DBMSs ■ Installing Asset Manager ■ Simple upgrade of Asset Manager version 4.2.x, 4.3.x or 4.4.x	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 Asset Manager	Text file	■ \doc\infos\database.txt ■ \doc\infos\tables.txt
		Online	\doc\chm\dbstruct*.chm
Structural differences between different database versions	❖ List of tables, fields, links and indexes that have changed.	Html	\doc\infos\diff*.html
Administering Asset Manager	■ Asset Manager Application Designer ■ 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), 77
!-- (migration.xml), 77
!--- (migration.xml), 77
^ (character), 31
' (special character), 77
" (special character), 77
& (special character), 77
< (special character), 77
> (special character), 77
| | (special character), 77

A

Access restrictions, 53
acmig.dtd, 83 , 70
Administration, 88
Advanced use, 88
Analyze only, 46
Application data
  (See Also Structure - propagating changes)
  Conversion file, 82
  Conversion rules, 43
  Definition, 65
Asset Manager Application Designer
  Database - copy, 37
  Database integrity - verification, 45 , 27
  Structural parameters - propagation, 86
  Structure - propagating changes (See Structure - propagating changes)
Asset Manager Automated Process Manager, 59
Asset Manager clients, 61
Asset Manager programs - updating, 59-61
  Process, 60
Asset Manager Web, 57
Assets, 83
Available guides, 87

B

Brands, 44 , 44

C

Caches, 60
Check validity of records (option), 46 , 29 , 28
config (folder), 55 , 52
Connections, 60
continueonerror (migration.xml), 70
Conversion
  Conflicts, 44
  Conversion machine, 24

Definition, 64
Final conversion, 49
Process, 40
Conversion file
Adapting, 67 , 38
Application data, 82
Assets, 83
Attributes
continueonerror, 70
encoding, 69
engine, 70
SYSTEM, 70
Conversion rules, 68
Definition, 64
Elements
!--, 77
!---, 77
Exception, 75
Field, 74
Mapping, 72
MigrationFile, 70
PostActions, 76
PreActions, 73
Script, 76
StartScript, 71
Translate, 71
Foreign keys, 81
Joins, 80
Multiple destination tables, 81
Numeric fields, 81
Special characters
;, 77
', 77
", 77
\, 77
&, 77
<, 77
>, 77
||, 77
SQL statements, 82
Syntax, 69
Verification, 83
What it is used for, 68
Conversion machine
Definition, 64
Preparation, 24
Conversion speed, 25
Converting the database, 38
Convert the database (menu), 40
Copy database to empty database (menu), 37
Counters, 29

D

Database
Blocking and copying, 49
Conversion
(See Also Conversion)
Copy, 36
Asset Manager Application Designer, 37
DBMS tools, 37
Traditional backup - problems, 36
Finalization, 50
Integrity - verification, 45 , 27
Manual adjustment, 29
Production phase, 61
Structure, 88
Structure - differences between versions, 84
Unsupported DBMSs, 18
Validation, 46
Database structure
Definition, 65
Data - definition, 65
dbbscripts (folder), 36
DBMS
Preparing the server, 26
Supported DBMSs, 15
Unsupported versions, 18
DBMS server - preparation, 26
Default values, 42
Definitions
Application data, 65
Conversion file, 64
Conversion machine, 64
Converting the Asset Manager database, 64
Data, 65

Migration, 63
Production database, 64
Structure of the Asset Manager database, 65
Trigger, 65
Updating Asset Manager programs, 63
diff*.*, 84
diff*.htm, 85
diff*.txt, 84
diff*.xml, 86
dtd (file extension), 70

E

encoding (migration.xml), 69
engine (migration.xml), 70
Exception (migration.xml), 75
External programs
Updating, 57
External tools - integration, 47

F

Features, 79 , 78 , 54
Field (migration.xml), 74
Fields
Dividing between several tables, 77
Fields storing application data to be converted manually, 43
Length, 31
Mandatory fields, 41 , 30
Numeric fields, 81
Transferring a feature, 78 , 54
Floor plan positions, 41
Foreign keys, 81
fromxxx (folder), 64 , 38
Functional rights, 53

G

gbbase.xml, 66 , 55 , 52
gbbase*.*, 86 , 41 , 33 , 30
Glossary, 63-66

H

Help on fields, 51
HP Connect-It, 57
Warning, 17
HP Connect-It scenarios, 57

I

Indexes, 51 , 42
infos (folder), 84
Installation, 88
Integrity - verification, 45

J

Java - installation, 25
Java Runtime - installation, 25
Joins, 80

M

Mandatory links, 30
Mapping (migration.xml), 72
Migration
Definition, 63
Final conversion, 49-50
Final phase, 59-62
Preparation phase
Preparatory phase, 21-26
Process, ?-19
Simulation, 27-47
migration.xml, 64 , 50 , 46 , 29
(See Also Conversion file)
MigrationFile (migration.xml), 70

N

newdbb.log, 35

O

Open (menu), 28 , 28
Open existing database (menu), 40 , 39 , 35
Owner (field), 40 , 39

P

PostActions (migration.xml), 76
PreActions (migration.xml), 73
Procurement (module), 32 , 18
Production database, 64

Propagate the customized structure (menu), 35

R

Readme, 87
References, 67-88
Release Notes, 87
Repair (option), 29 , 28
Repair database (menu), 28 , 28
Rollback segments, 26

S

SAP Crystal Reports, 54 , 54
Script (migration.xml), 76
sdu.log, 40
sdu.xml, 83
Special characters, 77
SQL statements, 82
SQL - statements, 82
StartScript (migration.xml), 71
Stored procedures, 51
Structural parameters, 41
  Propagation, 86
Structure of the database
  Changes - propagation (See Structure - propagating changes)
Structure - propagating changes, 33
  Conflicts, 36
  Overview, 34
  Process, 35
Supported environments
Supported operating systems, 15
SYSTEM (migration.xml), 70

T

Tables
  Conversion order, 42
  Dividing the fields between several tables, 77
  Multiple destination tables, 81
Training, 23
Transaction logs, 26
Translate (migration.xml), 71
Triggers, 51
  Definitions, 65

U

up_GetCounterVal (stored procedure), 51 , 29
Update, 63
User (field), 40 , 39
User rights, 53

V

Value (attribute), 42
Views, 51

W

Workflow (module), 32

X

xerces.jar, 36
xml (file extension), 70 , 36
XML - editor, 25
XML editor - installation, 25