

Peregrine

AssetCenter



Automatic software mechanisms

© Copyright 2005 Peregrine Systems, Inc.
All Rights Reserved.

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

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

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

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

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

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

This edition applies to version 4.4 of the licensed program

AssetCenter

Peregrine Systems, Inc.
3611 Valley Centre Drive San Diego, CA 92130
858.481.5000
Fax 858.481.1751
www.peregrine.com



Table of Contents

Introduction	11
Who is this guide intended for?	11
What does this guide offer?	11
How to use this guide	12
Chapter 1. Overview	13
Concepts linked to automatic mechanisms	13
Chapter 2. Presentation of the automatic mechanisms	17
Categories of automatic mechanisms	17
Definition of automatic mechanisms	18
Overview	21
Chapter 3. Automatic mechanisms in AssetCenter Server	23
Overview of AssetCenter Server	23
AssetCenter Server modules	24
Chapter 4. Assets table (amAsset)	37
Scripts	37
Integrity rules	47
Agents	49
Chapter 5. Assets Included in Projects table (amAstProjDesc)	61

Scripts	61
Chapter 6. Brands table (amBrand)	63
Scripts	63
Integrity rules	64
Agents	65
Chapter 7. Catalogs table (amCatalog)	67
Scripts	67
Integrity rules	68
Agents	69
Chapter 8. Products table (amCatProduct)	71
Scripts	71
Agents	75
Chapter 9. Catalog References table (amCatRef)	77
Scripts	77
Integrity rules	79
Agents	79
Chapter 10. Companies table (amCompany)	81
Scripts	81
Chapter 11. Computers table (amComputer)	83
Scripts	83
Integrity rules	87
Agents	87
Workflows	90
Chapter 12. Contacts table (amContact)	93
Scripts	93
Chapter 13. Contracts table (amContract)	95
Scripts	95
Integrity rules	108
Agents	109
Chapter 14. Cost Centers table (amCostCenter)	115
Scripts	115

Integrity rules	116
Agents	117
Chapter 15. Departments and Employees table (amEmplDept)	119
Scripts	119
Integrity rules	123
Agents	124
Chapter 16. Locations table (amLocation)	127
Scripts	127
Integrity rules	129
Agents	129
Workflows	130
Chapter 17. Models table (amModel)	133
Scripts	133
Agents	141
Chapter 18. Portfolio Items table (amPortfolio)	143
Scripts	143
Integrity rules	149
Agents	150
Workflows	163
Chapter 19. Projects table (amProject)	165
Scripts	165
Agents	166
Chapter 20. Stocks table (amStock)	167
Scripts	167
Chapter 21. Third-Party Companies table (amThirdParty)	169
Integrity rules	169
Agents	170
Chapter 22. Glossary	171
Database terms	171
A. Extracting all the scripts from a database	173
Executing a template in AssetCenter Database Administrator	174

Examples of templates	174
B. Determining the workflows used for a table	183
C. Extracting the list of fields and links of the screens	187
Executing a template in AssetCenter Database Administrator	188
Template example	188
Index	191



List of Figures

1.1. Database access	14
1.2. Sequence within a transaction	15
2.1. Positioning of automatic mechanisms	21



List of Tables

PEREGRINE

4.1. Validity scripts on the table	38
4.2. Default value scripts	38
4.3. Read-Only scripts	41
4.4. Irrelevance scripts	42
5.1. Default value scripts	62
6.1. Default value scripts	64
6.2. Mandatory scripts	64
7.1. Default value scripts	68
7.2. Irrelevance scripts	68
8.1. Validity scripts on the table	72
8.2. Default value scripts	72
8.3. Irrelevance scripts	74
9.1. Default value scripts	78
9.2. Mandatory scripts	79
10.1. Default value scripts	82
10.2. Irrelevance scripts	82
11.1. Default value scripts	84
11.2. Irrelevance scripts	86
12.1. Default value scripts	94
13.1. Validity scripts on the table	96
13.2. Default value scripts	96
13.3. Read-Only scripts	102
13.4. Irrelevance scripts	103
14.1. Validity scripts on the table	116
14.2. Default value scripts	116

15.1. Default value scripts	120
15.2. Irrelevance scripts	121
16.1. Default value scripts	128
17.1. Default value scripts	134
17.2. Mandatory scripts	135
17.3. Irrelevance scripts	136
18.1. Validity scripts on the table	144
18.2. Default value scripts	144
18.3. Mandatory scripts	146
18.4. Read-Only scripts	147
18.5. Irrelevance scripts	147
19.1. Validity scripts on the table	166
19.2. Default value scripts	166
20.1. Default value scripts	168



Introduction

Who is this guide intended for?

This guide is intended for all enterprises using AssetCenter.

It is intended for engineers who require detailed information concerning the automatic data-processing mechanisms in AssetCenter:

- Database administrators.
- Those in charge of implementation or customization.

What does this guide offer?

This guide offers an overview of the different types of automatic mechanisms used in AssetCenter and gives an exhaustive listing of the different conditions governing these mechanisms. It also describes in detail the mechanisms associated with certain core tables in the database.

How to use this guide

Warning:

Broad and in-depth knowledge of AssetCenter is required to make proper use of this guide. In particular, mastery of the following areas is assumed: Database structure, Portfolio organization, Basic language and scripting.

Chapter Overview

This chapter explains the underlying principles related to automatic mechanisms in AssetCenter.

Read this chapter for an overview of automatic mechanisms.

Chapter Presentation of the automatic mechanisms

This chapter explains and categorizes all the automatic mechanisms operating in AssetCenter.

Chapter Automatic mechanisms in AssetCenter Server

This chapter presents the automatic mechanisms in AssetCenter Server.

Chapters Assets table (amAsset), Computers table (amComputer) and Portfolio Items table (amPortfolio)

These chapters detail every automatic mechanism in the tables concerned.

Chapter Glossary

This glossary contains the definitions of several key terms related to automatic mechanisms.

Appendix Extracting all the scripts from a database

This appendix explains how to extract all scripts from your database.

Appendix Determining the workflows used for a table

This appendix explains how to list all the workflows in a given table.



1 Overview

CHAPTER

AssetCenter uses a set of automatic mechanisms with three objectives in mind:

- 1 To maintain the structural and logical integrity of the data stored in the database. For example, integrity rules to maintain the relationship between the values of multiple fields.
- 2 To facilitate data entry. For example, scripted default values to populate certain fields automatically on creating a record.
- 3 To apply business rules globally or specifically. For example, workflows to trigger archival of past expense lines.

The use of the term **automatic mechanism** as used in this guide is large. It covers any sort of automatic modification to the database by a component of AssetCenter, triggered by an event (entering information in the user interface, updating a record, deletion of data by a workflow, etc.). All other external mechanisms outside of AssetCenter or its components, is not covered in this guide. This is the case, for example, of automatic mechanisms defined at the database level, such as triggers and stored procedures.

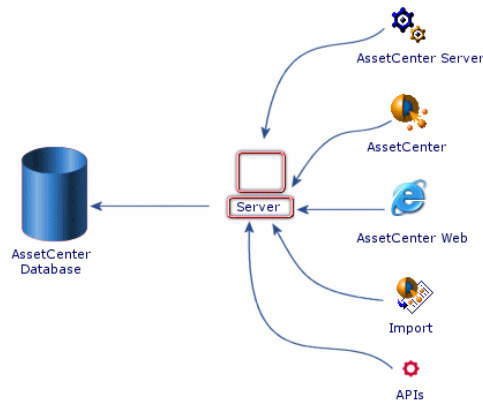
Concepts linked to automatic mechanisms

This section contains reminders of important general information concerning databases and specific information concerning the AssetCenter database.

Database access

The automatic mechanisms apply to all types of database access. The following diagram summarizes the different components that access that database:

Figure 1.1. Database access



Sequence of modification

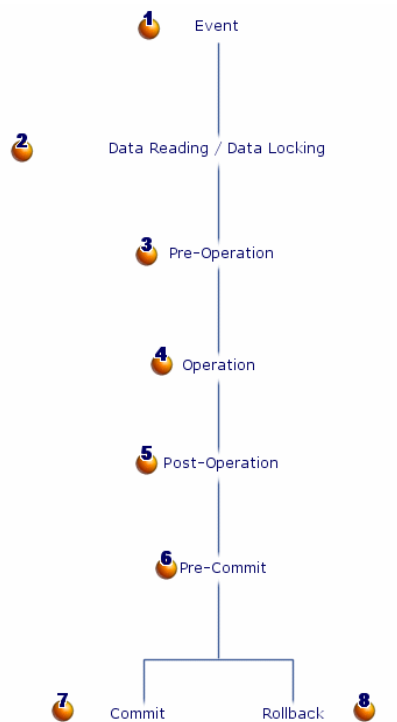
The modification of data in the database, whether it be an elementary operation (update, insert, delete) or a series of elementary operations, always follows the same sequence within a transaction.

 **Note:**

A transaction may be made up of several SQL queries. A database manipulation involving read and write operations may be consistent once finished but pass through intermediate stages that are not.

The typical sequence of modification is as follows:

Figure 1.2. Sequence within a transaction



- **1**: The event is not part of the transaction. It is at the origin of it. An event therefore means a manipulation that will potentially lead to a modification of the data in the database.
- **2**: In order to maintain the consistency of the transaction, a data-locking mechanism is used. In practice, the first transaction to use a data item locks it. The other transactions in progress may therefore not use it until it is unlocked.
- **3**, **4** and **5** constitute the sequential steps followed in all database operations be they INSERT, DELETE or UPDATE.

 Note:

There may be several operations and therefore several Pre-Operation / Operation / Post-Operation cycles within the same transaction.

- 🟡⁶ represents an interim step: The operations have been performed but the modifications have not yet been committed to the database.
 - 🟢⁷: The modifications have been committed to the database.
 - 🟡⁸: All the modifications have been cancelled. The database has not been modified by the transaction.
-

 Note:

Each database engine has its individual characteristics, in particular with regard to **Rollback** operations. Refer to the documentation provided with your DBMS for more information.



2 | Presentation of the automatic mechanisms

CHAPTER

Several automatic mechanisms are used in AssetCenter:

- Scripts
- Integrity rules
- Agents
- Synchronous and asynchronous workflows
- Automatic mechanisms handled by AssetCenter Server

The objective of this chapter is to provide you with the most exhaustive list possible of these automatic mechanisms.

Categories of automatic mechanisms

As a convention, we have chosen to classify the different automatic mechanisms in AssetCenter using three major groups. The categories depend on the persistence of the automatic mechanisms:

- 1 Permanent automatic mechanisms, as their name suggests, are permanently activated for all database access methods (Windows client, APIs, etc.) as well as at the transaction level. Scripts and integrity rules enter into this category.

- 2 Synchronous automatic mechanisms, which are only triggered as the result of an event (modification of a record, specific step in a transaction, etc.). Agents and asynchronous workflows enter into this category.
- 3 Asynchronous automatic mechanisms, which are triggered in an uncorrelated manner with reference to events. This category includes automatic mechanisms managed by AssetCenter Server and asynchronous workflows, which are not dealt with in detail in this chapter. For a complete description of workflows, refer to the **Advanced use** guide, **Workflow** chapter.

Definition of automatic mechanisms

Basic scripts

In AssetCenter, Basic scripts are used to define and control automatic behavior. AssetCenter ships with a standard set of predefined scripts (and automatic mechanisms). The administrators and users may create their own scripts.

Scripts work:

- at the record level, or
- at the field and link level

The following table summarizes the different types of scripts.

Script name	Field of application	Definition
Validity	Record	This script applies to all records in a table and makes it possible to define conditions for validating new or modified records. For example, you can define an automatic mechanism to forbid the creation of numeric type features if the maximum value is less than the minimum value.
Historized	Field or Link	This script enables you to define conditions for historizing modifications made to a field or link.

Script name	Field of application	Definition
Read only	Field or Link	This script enables you to define the conditions under which a field or link can be modified.
Mandatory	Field or Link	This script enables you to define the conditions making a field or link mandatory.
Default	Field or Link	This script enables you to define the value that is automatically proposed for a field or a link when a new record is created.
Irrelevance	Field or Link	This script conditions whether a field or a link is displayed.

Integrity rules

AssetCenter permanently checks the consistency between certain field before authorizing insert or update operations in the database.

In practice, an integrity rule is made up of three elements:

- 1 The list of monitored objects (fields and links)
- 2 The rule concerning the objects monitored to be verified
- 3 The list of objects (fields and links) that can be modified in order to check the rule

Warning:

An integrity rule constantly checks the rule for which it is created. It sometimes has to perform arbitrations and modify values to maintain database integrity.

The integrity rules work recursively. For example, if an integrity rule, A, triggered by the modification of a field, C, modifies a field D, which in turn is monitored by a second integrity rule, B, then integrity rule B will execute when field D is modified without waiting for rule A to finish working.

Agents

An agent is an automatic mechanism that is triggered at the same time as a transaction. This can be before (**Pre**), during or after (**Post**) one of the following operations:

- Insert
 - Update
 - Delete
-

 **Note:**

An agent can also be triggered before the database **Commit** operation.

An agent is made up of three elements:

- 1 The list of objects (fields and links) monitored by the agent with for each object the step of the transaction during which it is monitored.
- 2 The list of operations performed by the agent.
- 3 The list of objects (fields and links) updated by the process.

Agents work in cooperative mode. They are triggered once only and declare beforehand which objects are going to be modified by the process, thus allowing other agents to work.

Synchronous workflows

A synchronous workflow is a specific type of workflow used to implement behaviors that do not exist by default in AssetCenter. Unlike agents and integrity roles, workflows can be created and modified by the user. They are particularly suited to the needs of implementers who require company-specific or line-of-business-specific automatic mechanisms. In this type of workflow, events are processed immediately and the appropriate transitions are activated by AssetCenter Server.

For example, a synchronous workflow may be used to automatically propagate a changed cost center at the location level to the sub-locations.

There is no major functional difference between a synchronous workflow and an agent. Only their nature differs: An agent is hard coded in AssetCenter and cannot be modified, a synchronous workflow is part of the data in the database and may not be modified at will. Additionally, synchronous workflows are only executed after one of the operations previously mentioned (Insert, Update, Delete).

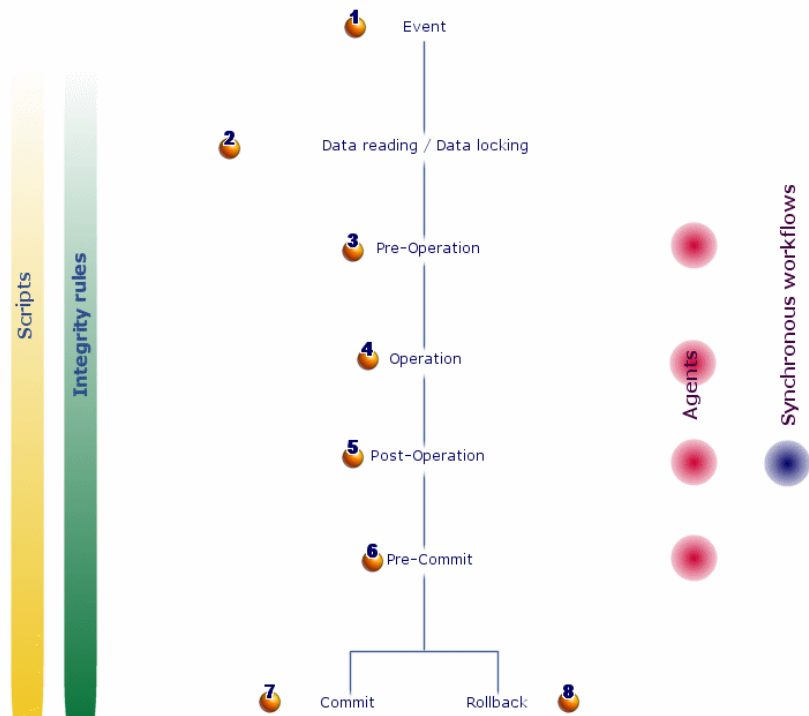
 Note:

We invite you to read the documentation on workflows, the **Advanced use guide**, **Workflow** chapter.

Overview

The following diagram gives an overview of how the different mechanisms fit together to modify data.

Figure 2.1. Positioning of automatic mechanisms





3 | Automatic mechanisms in AssetCenter Server

CHAPTER

This chapter includes reminders of the automatic mechanisms processed by AssetCenter Server.

 Note:

For further information, refer to the **Administration** guide, AssetCenter Server chapter.

Overview of AssetCenter Server

AssetCenter includes a system to monitor deadlines and automatically trigger actions: This program, called AssetCenter Server, operates independently of AssetCenter and automatically monitors all designated database deadlines. In particular:

- Alarms (end of term dates of contracts for example).
- Purchase request approvals.
- Stock line reorder levels.
- Rent calculations at the asset and the contract level.
- Lease contract loss value calculations.

- Expense line split operations associated with cost centers.
- Verification of history lines.
- Workflow deadlines.
- Searches for new workflow execution groups.
- Execution of workflow rules.
- Verification of time zones.

If justified to do so by the deadlines, AssetCenter Server performs actions, such as issuing reminder messages in the AssetCenter database via the internal messaging system. If necessary, it calculates contract rent, lease contract loss-values, etc.

Each automatic mechanism carried out by AssetCenter Server is defined as a module.

AssetCenter Server modules

Add the computers listed in the NT domain to the database module (AddCpu)

AssetCenter Server enables you to program the recovery of those computers declared in the NT domain.

The domain to analyze is specified at the Connect-It **addcpu . scn** scenario.

Add NT users to the database module (AddUser)

AssetCenter Server enables you to program the recovery of the users declared on the NT domain.

This is essentially used to populate the **Departments and employees** table with the information useful for connecting to an AssetCenter database that uses integrated NT security.

The domain to analyze is specified at the Connect-It **adduser . scn** scenario.

Calculate rents module (Rent)

AssetCenter Server monitors periodic rent payments for contracts and assets. It calculates and/recalculates the amounts involved.

The **Calculate rents** module defines:

- Certain parameters concerning the generation of costs for contracts and asset-level rent payments.
- The frequency of updates.

Overview

AssetCenter Server verifies at regular intervals whether it needs to generate expense lines. If this is so, it generates them.

After checking and generating the expense lines relative to a periodic rent, AssetCenter Server stores the date of the last expense line (past or present) in the **Recalculation effective from** field (SQL name: dRecalcul).

- If the contract-level rent is distributed to the assets, AssetCenter Server modifies the **Recalculation effective from** field that is found in the rent sub-tabs of the **Acquis.** tab of the assets detail.
- If the contract-level rent is not distributed to asset level, AssetCenter Server modifies the **Recalculation effective from** field, which is found in the rent sub-tabs of the **Rents** tab of the contract detail.

AssetCenter Server does not recalculate every single expense line each time.

- Projected expense lines associated with a periodic rent are always recalculated.
- The **Recalculation effective from** field, proper to each rent, sets the date from which past and present expense lines associated with a periodic rent are recalculated.

The lessee may directly modify the recalculation date of the non-projected expense lines by directly modifying the **Recalculation effective from** field. This flexibility enables you to recalculate erroneous expense lines in case of a change in tax rates, for example.

Parameters

The **User data item** field is used to set the rent generation parameters. The syntax of this field is as follows:

```
<Duration>j
```

This duration set the number of days for which the calculation is made. For example, if you want to calculate rent over a period of 90 days, enter the following value:

90d

 **Note:**

The maximum number of rent calculations made per transaction is specified by the `UserData` entry in the `Amsrv.cfg` configuration file.

Location of this file: ► **AssetCenter - Installation** guide, chapter **.ini and .cfg files**.

Projected rent

The **User data item** field enables you to specify the number of days for which you calculate project rent payments.

AssetCenter Server generates the projected expense lines for the specified period. In order to not generate any, you just need to set this field to **0**.

Example

Let's consider the following configuration:

- The contract is effective from July 1, 2001 through July 1, 2004.
- The rent is payable monthly on the first day of the month.
- AssetCenter Server verifies rent payments every 2 months and generates projected rent payments for the next 12 months.

On July 1, 2002, AssetCenter Server is launched for the first time: it generates:

- Past rents from July 1, 2001 through July 1, 2002.
- The present rent on July 1, 2002.
- The projected rents from August 1, 2002 through July 1, 2003.

Following these calculations, the `Recalculation effective from` field indicates the date of the last non-projected expense line, i.e. July 1, 2002.

AssetCenter Server runs in the background: 2 months later on September 1, 2002, it generates:

- The projected rents from October 1, 2002 through September 1, 2003.
- Past or present rents for which the payment date is later than that contained in the `Recalculation effective from` field, i.e. the rents from August 1, 2002 through October 1, 2002.

Calculate stipulated loss values module (LostVal)

AssetCenter Server recalculates, at regular intervals, the loss values for lease contracts whose calculation method is set to **Calculate for all periods** (Calculation field (SQL name: seLossValCalcMode) in **Leasing** tab of the lease contract detail). In this way, loss values pertaining to any loss value rules that have been modified since the last time AssetCenter Server accessed the database are updated.

Create assets, consumables, etc. corresponding to items received module (Delivery)

Prerequisites

This module cannot be executed unless you have already done the following:

- Execute AssetCenter.
- Select the **Administration/ Database options** menu.
- Select the **Procurement/ Let AssetCenter Server create the items received in the portfolio** option.
- Set this option to **Yes**.

Task performed by the module

This module is used to process the records from the **Items received** table in order to create received items (assets, consumptions, etc.) in their respective tables.

Utility of this mode

Assigning this task to AssetCenter Server rather than the AssetCenter application can increase the performances of those users receiving orders.

Frequency of execution

We recommend that you execute this module several times a day if you want the users to be able to quickly access the items received in their respective tables.

Execute workflow rules for execution group modules

Once a workflow execution group (Example: **ADMIN**) is detected, AssetCenter Server executes the appropriate workflow rules.

Monitoring of workflow execution groups

AssetCenter Server monitors the deadlines specific to workflow instances associated with the execution group.

Deadlines to be monitored by AssetCenter Server as soon as the activity is triggered are defined in the **Alarms** tab of the detail of the workflow activity.

These deadlines are defined by the time limits defined for the set tasks to be carried out.



In the case of deadlines specific to workflow, business calendars specified in the **Time limit** tab in the activity detail are taken into account. When calculating deadlines, these time limits are converted to business hours.

Processing of Periodical type events

According to the frequency defined in the **Parameters** tab in the detail of a **Periodical** type event, AssetCenter Server triggers the event if the activation conditions are met.

Then, the role of AssetCenter Server depends on the event's processing mode as indicated in the **General** tab of the event detail:

- **Log event and process by server:** As soon as the event occurs, AssetCenter Server saves it to the table with SQL name "amWfOccurEvent".
Then, AssetCenter Server activates the transition according to the frequency of verification as defined in the configuration screen of AssetCenter Server.
- **Log event and process immediately:** As soon as the event occurs, AssetCenter Server saves it to the table with SQL name "amWfOccurEvent", and activates the transition.
- **Process event immediately without logging:** As soon as the event occurs, AssetCenter Server activates the transition.

Activation of transitions

AssetCenter Server activates the transitions for events according to the frequency defined in the configuration screen. The following events are concerned:

- **System** events.
- **Database** and **Periodical** type events whose processing mode is set to **Log event and process by server**.

Execution of tasks

AssetCenter Server executes tasks resulting from **Automatic action** or **Test / script** type activities, except in the possible case of tasks resulting from activities whose **Execute actions immediately** (SQL name: bExecImmediately) box is selected.

The frequency with which AssetCenter Server verifies and performs the tasks it has to carry out is indicated in the configuration screen of AssetCenter Server.

In the case of a task originating from an **Automatic action** or **Test / script** type activity whose **Execute actions immediately** box (SQL name: bExecImmediately) is checked:

- This task is executed by AssetCenter Server if it is AssetCenter Server that activates the transition creating the task. In this case, AssetCenter Server performs the task as soon as the transition it creates is activated.
- Otherwise, the AssetCenter client machine executes the task.

Update the database with the results of the scanners module (IddAc)

AssetCenter Server enables you to program the recovery of the **.fsf** files produced by Desktop Inventory (these files store the results of the machine scan).

The folder containing the **.fsf** files is specified at the level of the Connect-It **iddac.scn** scenario.

 **Note:**

This module is based on the assumption that the machine scan has already been performed.

Update the database with the results of the scanners module (PdiAc)

This module enables you to program the recovery of `.xml.gz` files by Desktop Inventory. These files store the results of each computer scan.

The folder that contains the `.xml.gz` files is specified at the level of the Connect-It scenario `addcpu.scn`.

Update statistics for tables module (Stats)

This module updates the database statistics.

These statistics are used by all the DBMSs supported by AssetCenter to optimize SQL query plans.

If these statistics are not updated, the DBMS will not know which indexes are the most efficient.

We recommend that you execute this module once a week, or every night if your database is heavily modified.

Purge the input-events table module (PurgeEventInTable)

This module deletes the records from the **Input events** table according to the information in the:

- **Status** field (`seStatus`) of the **Input events** table (`amInputEvent`).
- **Deletion** field (`seStatus`) of the **Input events** table (`amInputEvent`).
- Expiration time defined by the **Events management/ Expiration time for input events (hours)**, accessible via the **Administration/ Database options** menu in the AssetCenter application.

Purge the outgoing-events table module (PurgeEventOutTable)

This module deletes the records from the **Input events** table according to the information in the:

- **Status** field (seStatus) of the **Output events** table (amOutputEvent).
- **Deletion** field (seStatus) of the **Output events** table (amOutputEvent).
- Expiration time defined by the **Events management/ Expiration time for output events (hours)**, accessible via the **Administration/ Database options** menu in the AssetCenter application.

Search for new workflow execution groups module (WorkflowFinder)

AssetCenter Server monitors the creation of new workflow execution groups. As soon as AssetCenter Server detects a new workflow execution group **G**, it creates a new monitoring module **Execution of workflow rules for execution group G**.

This mechanism has the following advantages:

- It enables you to define verification timetables for each workflow execution group.
- Different workflow execution groups can be monitored by different instances of AssetCenter Server.

Signal presence of database server module (UpdateToken)

AssetCenter Server regularly sends a signal to the database server in order to indicate that it is functioning.

If the database server does not receive a signal from AssetCenter Server for over one hour, a message is displayed when a user connects to the database in AssetCenter.

This message indicates that AssetCenter Server has not been launched on this database for over one hour and that without this process, monitoring functions will be interrupted.

If the database server goes without receiving a signal from AssetCenter Server for over a week, it is no longer possible to connect to the database.

Split expense lines in cost centers module (CostCenter)

AssetCenter Server handles split operations for expense lines.

General overview

AssetCenter Server searches the expense lines to be split: These are the expense lines whose **Split operation status** field (SQL name: seSplitStatus) is set to **Not split**.

By default, all expense lines are to be split, regardless of their status (**Status** field (SQL name: seStatus) of an expense line).

AssetCenter Server splits the designated expense lines. When an expense line is split:

- A debit expense line, equivalent to the split expense line is created in the parent cost center.
- Expense lines are created in the target cost centers, according to the split percentage values. By default, these are **Not split**.

Specific example: Managing the removal of a cost center

When you decide to delete a cost center, and the cost center contains expense lines, AssetCenter will not allow you to perform the operation unless the **Authorize extended deletions** option in the **Edit** category of the **Edit/ Options** menu is checked.

In this case, AssetCenter gives you three possibilities:

- Delete all the linked records.
- Detach the linked records.
- Attach the linked records to another record.

What happens next depends on the option you choose:

Delete all linked records

When a cost center is deleted, AssetCenter deletes:

- The expense lines of the deleted cost center.
- The expense lines resulting from split operations on the deleted cost center.

An AssetCenter agent modifies the **Split operation status** field (seSplitStatus) so it displays "Not split" at the level of the expense lines highest up in the split operation. When these high-level expense lines were split, they generated the expense lines belonging to the deleted cost center (after any intermediate split operations).

When AssetCenter Server finds these expense lines, which are not split but have generated split expense lines, it deletes all the expense lines resulting from their split operations. In doing this, AssetCenter Server deletes the expense lines that, when split, generated the expense lines belonging to the deleted cost center.

Then AssetCenter Server performs the split operations on those expense lines, which have not yet been split. It thus recalculates, using new parameters, all the expense lines that, when split, generated the expense lines of the deleted cost center.

Detach all linked records

In this case:

- The expense lines of the deleted cost center are no longer associated with a cost center.
- The expense lines, which when split generated the expense lines for the deleted cost center, are split again.
- The expense lines, resulting from split operations on the deleted cost center, are not modified.

Attach linked records to another record

In this case, you select another cost center X, which takes the place of the deleted cost center:

- The expense lines of the deleted cost center are attached to cost center X.
- The expense lines, which when split generated the expense lines for the deleted cost center, are split again; cost center X is considered as the new target cost center.
- The expense lines resulting from split operations on the deleted cost center are deleted and the expense lines of cost center X are split.

Verify database server time zone module (TimeZone)

This module verifies the delay between the local time of the server and the client machines. This is useful if you specified a time zone for a client machine (menu **Administration/ Time zones**).

Verify alarms module (Alarms)

List of alarms monitored

At the asset level

Several key dates are monitored:

- The end-of-reservation date of an asset: This is shown in the **Reserv. end date** field (SQL name: dtEnd) in the **Portfolio/Reservations** tab of the asset detail.
- The warranty expiration date of an asset: Asset detail, **Maint. tab, Expiration** field (SQL name: dWarrEnd).
- End-of-term date for lease, rental, loan of an asset: This alarm can only be defined if the acquisition method of the asset (Asset detail, **Acquis. tab, Acq. method** field (SQL name: seAcquMethod)) is set to **Lease, Rental** or **Loan**. In this case the **Price and conditions** sub-tab of the **Acquis. tab** shows an **End date** field (SQL name: dEndAcqu).
- End-of-rent dates of an asset: Alarms can be attached to end of validity dates (**Acquis. tab, rent descriptions** sub-tabs, **Schedule** frame).

At the consumable level

AssetCenter Server monitors the end-of-reservation date for consumables: This is shown in the **Reserv. end date** field (SQL name: dReservEnd) in the reservation detail of a consumable. To access the reservation detail of a consumable:

- 1 Launch AssetCenter.
- 2 Select **Procurement/ Purchase requests**.
- 3 Select the purchase request reserving the consumable.
- 4 Display the composition of this purchase request.
- 5 Display the request line corresponding to the consumable.
- 6 Display the **Reservations** tab of the request line. This tab shows the list of reservations for consumables.
- 7 Display the detail of the reservation.

The monitored field is **Date fin** (SQL name: dtEnd).

At the project level

AssetCenter Server monitors the end dates of project: Project detail, **General** tab, **End** field (SQL name: dEnd).

At the contract level

Several key dates are monitored:

- The end-of-term date: Contract detail, **General** tab, **End** field (SQL name: dEnd).
- If the contract **Type** (SQL name: seType) is **Lease schedule** or **Master lease**: Alarms can be attached to the notification dates for possible end of lease. These dates are shown to the right of the notification field in the sub-tabs describing the possible end of term options: **Renewal**, **Purchase**, **Return**.
- If the contract **Type** (SQL name: seType) is **Lease schedule**: Alarms can be attached to the end dates of validity for rent items as shown in the individual rent-description sub-tabs of the **Rent** tabs.

At the purchase request level

If the acquisition method of the purchase request (Purchase request detail, **Financing** tab, **Acq. method** field (SQL name: seAcquMethod)) is set to **Lease**, **Rental** or **Loan**, it is possible to define an alarm associated with the rental, lease or loan end dates (**Acq. method** field in **Financing** tab of purchase request detail). The same is true for estimates and orders.

What happens in two-level alarms when the first level action has been triggered?

In the case of alarms with 2 levels, the triggering of the second level alarm depends on the action carried out at the first level.

- If the first-level alarm triggers an action other than the sending of a message via AssetCenter's internal messaging system (such as sending a message via a third-party messaging system), the second-level alarm will always be triggered at the defined moment.
- If the first level-alarm sends a message to a group of AssetCenter users via the internal messaging system, the action defined at the second level will not be triggered if one or more of the recipients has read the message.

Verify null-identifier records module (History)

This module verifies integrity of the records whose primary keys are null.

These records are automatically created in all the tables when the database is created.

They are used by AssetCenter to perform certain administrative tasks (which is transparent to you).

This module verifies that these records still exist, and will recreate them if necessary.

We recommend that you execute this module at least once every day to maintain the integrity of the database.

Verify history lines module (History)

Sometimes when a record is destroyed in the database, the corresponding history lines are not destroyed. AssetCenter Server verifies if there are any such history lines; if it finds any it destroys them.

Verify stocks module (Stock)

AssetCenter Server monitors stock reorder levels.

For each stock, AssetCenter Server refers to the stock rules defined in the **Manage** tab of the stock detail.

For each stock rule concerning a model:

- AssetCenter Server calculates the quantity of items actually available from the **Assignments** field in the detail of a portfolio item.
- When the quantity falls below the value specified in the **Reorder level** (SQL name: lReordLevel) field of the stock rule detail, AssetCenter Server automatically creates a purchase request.
 - The parameters of the purchase request can be found in the **Auto-request** and **Management** tabs of the detail of the stock.
 - The purchase request specifies the quantity to be reordered (**To order** field (SQL name: lQtyToOrder) in the detail of the stock rule).
- For as long as the request is not fully received, AssetCenter Server does not verify the stock rule that it has generated. Therefore, no new request is sent.
- As soon as delivery of the request is taken in full, AssetCenter Server:
 - Readjusts the stock levels.
 - Erases the contents of the **Request line** field (SQL name: ReqLine) in the stock rule detail.
 - Reactivates the stock rule.



4 Assets table (amAsset)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Portfolio Items table. Each section deals with a different type of automatic mechanism.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 4.1. Validity scripts on the table

Script	Description
<pre>If Not IsEmpty([dEndAcqu]) and Not IsEmpty([dStartAcqu]) and [dStartAcqu] > [dEndAcqu] Then Err.Raise(-2009, "The end date (dEnd) must be greater than or equal to the start date (dStart).") RetVal = FALSE Else RetVal = TRUE End If</pre>	<p>If the acquisition start date of the asset comes after the acquisition end date, the record is rejected.</p>

Table 4.2. Default value scripts

Object concerned	Script	Description
AcctCode	<code>RetVal = [Model.AcctCode]</code>	By default, the accounting code of the asset is that of the model.
AssetTag	<code>RetVal = [Model.Prefix] + AmCounter("amAsset_AssetTag", 6)</code>	By default, the asset tag of the asset is the concatenation of the prefix of the model and the value of the amAsset_AssetTag counter on 6 figures.
dAcquisition	<code>RetVal = [dStartAcqu]</code>	By default, the purchase date is set to the start of lease, loan or rental date.
dDeprRecalc	<code>RetVal = AmDate()</code>	By default, this field is set to the current system date.
DeprBasisCur	<code>RetVal = [PriceCur]</code>	By default, the currency in which the depreciation basis of an asset is expressed is identical to the one used to express its purchase value.
DeprValCur	<code>RetVal = AmDefaultCurrency()</code>	By default, this field is set to the value of the default currency.
dInstall	<code>RetVal = AmDate()</code>	By default, this field is set to the current system date.
dStartAcqu	<code>RetVal = AmDate()</code>	By default, this field is set to the current system date.

Object concerned	Script	Description
dtDeprBasisCv	RetVal = AmDate()	By default, this field is set to the current system date.
dtDeprValCv	RetVal = AmDate()	By default, this field is set to the current system date.
dtIntPayCv	RetVal = AmDate()	By default, this field is set to the current system date.
dtIntPayTaxCv	RetVal = AmDate()	By default, this field is set to the current system date.
dtListPriceCv	RetVal = AmDate()	By default, this field is set to the current system date.
dtMarketValCv	RetVal = AmDate()	By default, this field is set to the current system date.
dtNetValueCv	RetVal = AmDate()	By default, this field is set to the current system date.
dtPaymentsCv	RetVal = AmDate()	By default, this field is set to the current system date.
dtPriceCv	RetVal = AmDate()	By default, this field is set to the current system date.
dtPurchOptValCv	RetVal = AmDate()	By default, this field is set to the current system date.
dtResalePriceCv	RetVal = AmDate()	By default, this field is set to the current system date.
dtTaxCv	RetVal = AmDate()	By default, this field is set to the current system date.
fTotalQty	RetVal = 1	By default, the total quantity of the batch is 1.
IntPayCur	RetVal = AmDefaultCurrency() ()	By default, this field is set to the value of the default currency.
IntPayTaxCur	RetVal = AmDefaultCurrency() ()	By default, this field is set to the value of the default currency.
Label	RetVal = [AssetTag]	By default, the label of an asset is set to the asset tag. This is only relevant when the asset is a cable device.
IDeprSchId	RetVal = [Model.IDeprSchId]]	By default, the depreciation type of an asset is that of its model.

Object concerned	Script	Description
IIconId	RetVal = [Model.IIconId]	By default, this field, which contains the identifier of the icon used to represent the asset, inherits the same value as that of the model from which it is derived.
ListPriceCur	RetVal = AmDefaultCurrency ()	By default, this field is set to the value of the default currency.
ILabelRuleId	RetVal = [Model.ILabelRule Id]	By default, the label rule of an asset is set to the model. This is only relevant when the asset is a cable device.
ILessorId	RetVal = [PordLine.POrder. lSuppId]	By default, the lessor of an asset is the supplier of the purchase order line at the origin of the creation of the asset.
IModelId	RetVal=[PortfolioItem.lMod elId]	By default, the model associated with the asset is that of the associated portfolio item.
IPhotoid	RetVal = [Model.IPhotoId]	By default, the photo of the asset is that of the model.
ISoftLicUseRights	RetVal = [Model.ISoftLicUs eRights]	By default, the installation and utilization rights are that of the model.
ISuppId	RetVal = [PordLine.POrder. lSuppId]	By default, the supplier of an asset is the supplier of the purchase order line at the origin of the creation of the asset.
MarketValCur	RetVal = [PriceCur]	By default, the initial value of the asset is its purchase price.
mDeprBasis	RetVal = [mPrice]	By default, the depreciation basis of the asset is set to its purchase value.
mMarketVal	RetVal = [mPrice]	By default, the initial value of the asset is its purchase price.
NetValueCur	RetVal = AmDefaultCurrency ()	By default, this field is set to the value of the default currency.
PaymentsCur	RetVal = AmDefaultCurrency ()	By default, this field is set to the value of the default currency.

Object concerned	Script	Description
PriceCur	<code>RetVal = AmDefaultCurrency ()</code>	By default, this field is set to the value of the default currency.
PurchOptValCur	<code>RetVal = AmDefaultCurrency ()</code>	By default, this field is set to the value of the default currency.
ResalePriceCur	<code>RetVal = AmDefaultCurrency ()</code>	By default, this field is set to the value of the default currency.
sePeriodicity	<code>RetVal = 30</code>	By default, the frequency of associated rent payments is monthly (30 days).
seSoftLicMulti	<code>RetVal = [Model.seSoftLicMulti]</code>	By default, the software license type is that of the associated model.
seSoftLicType	<code>RetVal = [Model.seSoftLicType]</code>	By default, the software utilization license type is that of the associated model.
SoftMedia	<code>RetVal = [Model.SoftMedia]</code>	By default, the installation media is that of the associated model.
SoftOS	<code>RetVal = [Model.SoftOS]</code>	By default, the operating system is that of the associated model.
TaxCur	<code>RetVal = AmDefaultCurrency ()</code>	By default, this field is set to the value of the default currency.

Table 4.3. Read-Only scripts

Object concerned	Script	Description
fTotalQty	<code>RetVal = (2=[Model.Nature.seMgtConstraint] OR [ModelId]=0 OR [lAstId]<>0)</code>	

Table 4.4. Irrelevance scripts

Object concerned	Script	Description
dAcquisition	<code>RetVal = (0<>[seAcquMethod])</code>	This field, containing the acquisition date of the asset, is only relevant if the acquisition method of the asset is Purchase .
dDeprRecalc	<code>RetVal = (0<>[seAcquMethod])</code>	This field, containing the estimation date of depreciations and the residual value of the asset, is only relevant if the acquisition method of the asset is Purchase .
dEndAcqu	<code>RetVal = (1<>[seAcquMethod] AND 2<>[seAcquMethod] AND 3<>[seAcquMethod])</code>	This field, containing the end of acquisition date of the asset, is only relevant if the acquisition method of the asset is Rental, Lease, or Loan .
dIntPay	<code>RetVal = (1<>[seAcquMethod] AND 2<>[seAcquMethod])</code>	This field, containing the initial payment date of the asset, is only relevant if the acquisition method of the asset is Rental, Lease .
FixedAstNo	<code>RetVal = (0<>[seAcquMethod])</code>	This field, containing the fixed asset number of the asset, is only relevant if the acquisition method of the asset is Purchase .
Label	<code>RetVal = 1 ' Must be an asset and a device if [Model.Nature.seBasis] = 1 and [Model.Nature.bDevice] > 0 then RetVal = 0 end if</code>	This field, which contains the label of the asset, is only relevant if the asset is a cable device.
IAcquCntrlId	<code>RetVal = (1<>[seAcquMethod] AND 2<>[seAcquMethod] AND 3<>[seAcquMethod])</code>	The link to a rental or leasing contract is only relevant if the acquisition method is Rental, Lease or Loan .
Language	<code>RetVal = (0=[Model.Nature.bSoftLicense])</code>	This field, containing the language version of the software, is only relevant if the asset is a software item.

Object concerned	Script	Description
IDeprSchld	<pre>RetVal = (0<>[seAcquMethod])</pre>	The link to a depreciation type is only relevant if the acquisition method of the asset is Purchase .
LessorCode	<pre>RetVal = (1<>[seAcquMethod] AND 2<>[seAcquMethod])</pre>	This field, containing the lessor code, is only relevant if the acquisition method of the asset is Rental or Lease .
ILabelRuleId	<pre>RetVal = 1 ' Must be an asset and a d evice if [Model.Nature.seBasis] = 1 and [Model.Nature.bDe vice] > 0 then RetVal = 0 end if</pre>	This link to the label rule of the asset is only relevant if the asset is a cable device.
ILessorId	<pre>RetVal = (1<>[seAcquMethod] AND 2<>[seAcquMethod] AN D 3<>[seAcquMethod])</pre>	The link to a lessor is only relevant if the acquisition method is Rental, Lease or Loan .
ILicCntrId	<pre>RetVal = (0=[Model.Nature. bSoftLicense])</pre>	The link to a license contract is only relevant if the asset is a software item.
ISoftLicUseRights	<pre>RetVal = (0=[Model.Nature. bSoftLicense])</pre>	This field, containing the number of utilization or installation rights, is only relevant if the asset is a software item.
mDeprBasis	<pre>RetVal = (0<>[seAcquMethod])</pre>	This field, containing the depreciation basis of the asset, is only relevant if the acquisition method of the asset is Purchase .
mDeprVal	<pre>RetVal = (0<>[seAcquMethod])</pre>	This field, containing the depreciation value of the asset, is only relevant if the acquisition method of the asset is Purchase .
mIntPay	<pre>RetVal = (1<>[seAcquMethod] AND 2<>[seAcquMethod])</pre>	This field, containing the initial payment for the asset, is only relevant if the acquisition method of the asset is Rental, Lease .

Object concerned	Script	Description
mIntPayTax	<pre>RetVal = (1<>[seAcquMethod] AND 2<>[seAcquMethod])</pre>	This field, containing the total amount of taxes on the initial payment made when acquiring the asset, is only relevant if the acquisition method of the asset is Rental, Lease .
mListPrice	<pre>RetVal = (0<>[seAcquMethod] AND 1<>[seAcquMethod] AND 2<>[seAcquMethod])</pre>	This field, containing list price of the asset, is only relevant if the acquisition method of the asset is Purchase, Rental or Lease .
mNetValue	<pre>RetVal = (0<>[seAcquMethod])</pre>	This field, containing the residual value of the asset, is only relevant if the acquisition method of the asset is Purchase .
mPrice	<pre>RetVal = (0<>[seAcquMethod])</pre>	This field, containing the purchase price of the asset, is only relevant if the acquisition method of the asset is Purchase .
mPurchOptVal	<pre>RetVal = (2<>[seAcquMethod])</pre>	This field, containing purchase option value of the asset, is only relevant if the acquisition method of the asset is Lease .
mTax	<pre>RetVal = (0<>[seAcquMethod])</pre>	This field, containing the purchase price of the asset, is only relevant if the acquisition method of the asset is Purchase .
pDiscount	<pre>RetVal = (0<>[seAcquMethod] AND 1<>[seAcquMethod] AND 2<>[seAcquMethod])</pre>	This field, containing the standard discount price of the asset, is only relevant if the acquisition method of the asset is Purchase, Rental or Lease .
sCnxCount	<pre>RetVal = 1 ' Must be connectable if [Model.Nature.seBasis] = 1 and [Model.Nature.bIsCnxClient] > 0 then RetVal = 0 end if</pre>	This field is only relevant for cable devices.

Object concerned	Script	Description
seCnxStatus	<pre>RetVal = 1 ' Must be connectable if [Model.Nature.seBasis] = 1 and [Model.Nature.bIs CnxClient] > 0 then RetVal = 0 end if</pre>	This field is only relevant for cable devices.
seSoftLicType	<pre>RetVal = (0=[Model.Nature. bSoftLicense])</pre>	This link, containing the utilization license type, is only relevant if the asset is a software item.
SharingName	<pre>RetVal = 1 ' Must be connectable if [Model.Nature.seBasis] = 1 and [Model.Nature.bIs CnxClient] > 0 then RetVal = 0 end if</pre>	This field is only relevant for cable devices.
sMaxCnxCount	<pre>RetVal = 1 ' Must be connectable if [Model.Nature.seBasis] = 1 and [Model.Nature.bIs CnxClient] > 0 then RetVal = 0 end if</pre>	This field is only relevant for cable devices.
SoftMedia	<pre>RetVal = (0=[Model.Nature. bSoftLicense])</pre>	This field, containing the installation media, is only relevant if the asset is a software item.
SoftOS	<pre>RetVal = (0=[Model.Nature. bSoftLicense])</pre>	This link, containing the operating system, is only relevant if the asset is a software item.
TerminOpt	<pre>RetVal = (1<>[seAcquMethod] AND 2<>[seAcquMethod])</pre>	This field, containing the termination option of the rental or leasing contract, is only relevant if the acquisition method is Rental or Lease .
VersionLevel	<pre>RetVal = (0=[Model.Nature. bSoftLicense])</pre>	This link is only relevant if the asset is a software item.
AcquContract	<pre>RetVal = (1<>[seAcquMethod] AND 2<>[seAcquMethod] AN D 3<>[seAcquMethod])</pre>	The link to a rental contract is only relevant if the acquisition method is Rental , Lease or Loan .

Object concerned	Script	Description
AssetSlots	<pre>RetVal = 1 ' Must be an asset and a device if [Model.Nature.seBasis] = 1 and [Model.Nature.bDevice] > 0 then RetVal = 0 end if</pre>	This field is only relevant for cable devices.
DeprScheme	<pre>RetVal = (0<>[seAcquMethod])</pre>	The link to a depreciation type is only relevant if the acquisition method of the asset is Purchase .
FixedAssets	<pre>RetVal = ((0<>[seAcquMethod]) And (3<>[seAcquMethod]))</pre>	The link to the associated fixed assets is only relevant if the acquisition method of the asset is Purchase or Loan .
LabelRule	<pre>RetVal = 1 ' Must be an asset and a device if [Model.Nature.seBasis] = 1 and [Model.Nature.bDevice] > 0 then RetVal = 0 end if</pre>	This field is only relevant for cable devices.
Lessor	<pre>RetVal = (1<>[seAcquMethod] AND 2<>[seAcquMethod] AND 3<>[seAcquMethod])</pre>	The link to a lessor is only relevant if the acquisition method is Rental , Lease or Loan .
LicenseContract	<pre>RetVal = (0=[Model.Nature.bSoftLicense])</pre>	The link to a license contract is only relevant if the asset is a software item.
Link	<pre>RetVal = 1 ' Must be an asset and a device if [Model.Nature.seBasis] = 1 and [Model.Nature.bDevice] > 0 then RetVal = 0 end if</pre>	This field is only relevant for cable devices.
Pins	<pre>RetVal = 1 ' Must be an asset and a device if [Model.Nature.seBasis] = 1 and [Model.Nature.bDevice] > 0 then RetVal = 0 end if</pre>	This field is only relevant for cable devices.

Object concerned	Script	Description
Ports	<pre>RetVal = 1 ' Must be connectable if [Model.Nature.seBasis] = 1 and [Model.Nature.bIs CnxClient] > 0 then RetVal = 0 end if</pre>	This field is only relevant for cable devices.
Rents	<pre>RetVal = (1<>[seAcquMethod] AND 2<>[seAcquMethod] AN D 3<>[seAcquMethod])</pre>	The link to the rent payments is only relevant if the acquisition method is Rental, Lease or Loan .

Integrity rules

Name of the rule	List of monitored objects	Rule(s) verified	List of any modified objects
CAssetDeprInteg	<ul style="list-style-type: none"> ■ SyncRead on object: mDeprBasis ■ SyncRead on object: mDeprVal ■ SyncRead on object: mNetValue ■ SyncRead on object: DeprBasisCur ■ SyncRead on object: DeprValCur ■ SyncRead on object: NetValueCur 	<p>The rule forces the following relationship:</p> $mNetValue = DeprBasis - mDeprVal$ <p>The residual value of an asset is always equal to the depreciation basis minus all amortizations.</p>	<ul style="list-style-type: none"> ■ DeprValCur ■ mDeprVal ■ mNetValue ■ NetValueCur

Name of the rule	List of monitored objects	Rule(s) verified	List of any modified objects
CBiSoftInteg	<ul style="list-style-type: none"> ■ ASyncRead on object: Model.Nature.bSoftLicense ■ SyncRead on object: ISoftLicUseRights ■ SyncRead on object: seSoftLicType ■ SyncRead on object: seSoftLicMulti 	<p>The following rules are enforced for an asset for which the nature of the model is a software license:</p> <ul style="list-style-type: none"> ■ If the license is not Multiple-user (seSoftLicMulti), the number of users for the license (ISoftLicUserRights) is forced to 1. The license type (seSoftLicType) is forced to Per named workstation. ■ If the number of users of the license is greater than 1, the license becomes Multiple-user. 	<ul style="list-style-type: none"> ■ ISoftLicUseRights ■ seSoftLicMulti ■ seSoftLicType

Name of the rule	List of monitored objects	Rule(s) verified	List of any modified objects
CDeprScriptInteg	<ul style="list-style-type: none"> ■ SyncRead on object: mDeprBasis ■ SyncRead on object: dDeprRecalc ■ SyncRead on object: dStartAcqu ■ SyncRead on object: lDeprSchld ■ SyncRead on object: DeprBasisCur 	If one of the monitored objects is updated, the rule runs the depreciation calculation script.	

Agents

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CAssetPinAgent	<ul style="list-style-type: none"> ◆ Insert on object: amAsset 	This agent creates the pins/terminals for the asset depending on the specified number in the model.	
CAssetPortAgent	<ul style="list-style-type: none"> ■ Insert on object: amAsset ■ PreUpdate on object: lModelld 	This agent maintains the integrity of any connections between an asset and another asset.	
CAssetSlotAgent	<ul style="list-style-type: none"> ◆ Insert on object: amAsset 	This agent takes the list of slots defined in the model and creates the corresponding slots for the asset.	

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CBatchQtyAgent	<ul style="list-style-type: none"> ■ Insert on object: amPortfolio ■ PostUpdate on object: fTotalQty ■ PostUpdate on object: fQty ■ PostUpdate on object: lAstId ■ PreDelete on object: amPortfolio ■ PreDelete on object: amAsset 	This agent maintains the consistency between the total quantity of a batch (fTotalQty) and the sum of the quantities of the batch items (fQty).	
CComputeNextRntStepAgent	<ul style="list-style-type: none"> ◆ PostUpdate on object: dAccept 	This agent adjusts the rent recalculation date according to the acceptance date of the asset.	dRecalcul in the amAssetRent table.
CDateAlarmAgent	<ul style="list-style-type: none"> ◆ PostUpdate on object: dEndAcqu 	This agent recalculates if necessary the alarms associated with the end of acquisition date of an asset.	None in the amAsset table.
CDateAlarmAgent	<ul style="list-style-type: none"> ◆ PostUpdate on object: dWarrEnd 	This agent recalculates if necessary the alarms associated with the end of warranty date of an asset.	None in the amAsset table.
CDateAlarmAgent	<ul style="list-style-type: none"> ◆ PostUpdate on object: dEndCnx 	This agent recalculates if necessary the alarms associated with the end of connection date of an asset.	None in the amAsset table.

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CGbAcquiDepAgent	<ul style="list-style-type: none"> ■ Insert on object: amAsset ■ Insert on object: amPortfolio ■ PostUpdate on object: dAcquisition ■ PostUpdate on object: mPrice ■ PostUpdate on object: mTax ■ PostUpdate on object: dIntPay ■ PostUpdate on object: mIntPay ■ PostUpdate on object: mIntPayTax ■ PostUpdate on object: seAcquMethod 	<p>This agent updates the expense lines associated with the asset. It functions when an asset is created or the following data items are updated for an existing asset:</p> <ul style="list-style-type: none"> ■ seAcquMethod ■ dAcquisition ■ mPrice ■ mTax ■ mIntPay ■ mIntPayTax ■ dIntPay <p>Note:</p> <p>The agent takes into account the distribution (split-billing) of expenses to the cost centers and cost categories. It may therefore create multiple expense lines.</p>	None in the amAsset table.

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CGbAssetAssignment	<ul style="list-style-type: none"> ■ Insert on object: amCable ■ Insert on object: amContract ■ Insert on object: amComputer ■ Insert on object: amSoftInstall ■ Insert on object: amAsset ■ Insert on object: amPortfolio ■ Insert on object: amTraining ■ Insert on object: amWorkOrder ■ Insert on object: amPhone ■ PostDelete on object: amPortfolio ■ PreUpdate on object: IModelId ■ PreUpdate on object: IModelId ■ PreUpdate on object: IModelId ■ PreUpdate on object: IModelId ■ PreUpdate on object: IModelId 	In case of creation of a portfolio item, if necessary this agent creates the corresponding record in the Assets table and the appropriate record in the overflow table matching the management constraint associated with the model of the portfolio item.	
CGbAssetPriceAgent	<ul style="list-style-type: none"> ◆ PreUpdate on object: seAcquMethod 	If the acquisition method of the asset is not Purchase , the agent empties the Purchase date and Purchase price fields of the asset.	<ul style="list-style-type: none"> ■ dAcquisition ■ mPrice

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CGBienContratAgent	<ul style="list-style-type: none"> ■ Insert on object: amAsset ■ PostDelete on object: amAstCntr-Desc ■ PostUpdate on object: ICntrId ■ PostUpdate on object: IAsId ■ PostUpdate on object: IMaintCntrId ■ PostUpdate on object: IAcquCntrId 	<p>This agent maintains the synchronization of the data between the Contracts tab of an asset and the Schedule and Maint. contract fields in the asset detail:</p> <ul style="list-style-type: none"> ■ If one of these two fields is populated with a contract, then it is added to the list contracts in the Contracts tab. ■ If a rental or maintenance contract is removed from the list of contracts in the Contracts tab, the corresponding field is emptied. 	<ul style="list-style-type: none"> ■ Schedule ■ Maint. contract
CompteConnexions	<ul style="list-style-type: none"> ■ Insert on object: amPort ■ PostDelete on object: amPort ■ PostUpdate on object: IPortId ■ PostUpdate on object: IAsId ■ PreUpdate on object: sCnxCount 	<p>This agent counts the number of ports linked to the asset (number of items listed in the Ports tab of the asset detail) and stores the information in the sCnxCount field.</p>	<ul style="list-style-type: none"> ◆ sCnxCount

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CRedundancyAgent	<ul style="list-style-type: none"> ■ Insert on object: amAsset ■ Insert on object: amPortfolio ■ PostUpdate on object: AssetTag ■ PostUpdate on object: AssetTag ■ PostUpdate on object: IAstId 	<p>This agent makes sure that the AssetTag fields of an asset and its associated portfolio item are identical:</p> <ul style="list-style-type: none"> ■ If the AssetTag field of a record in the amAsset table is modified, the agent propagates this change to the AssetTag field of the record in the corresponding amPortfolio table. ■ If the AssetTag field of a record in the amPortfolio table is modified, the agent propagates this change to the AssetTag field of the record in the corresponding amAsset table. 	◆ AssetTag

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CRedundancyAgent	<ul style="list-style-type: none"> ■ Insert on object: amAsset ■ Insert on object: amPortfolio ■ PostUpdate on object: IModelId ■ PostUpdate on object: IModelId ■ PostUpdate on object: IAstId 	<p>This agent makes sure that an asset and its associated portfolio item always point to the same model:</p> <ul style="list-style-type: none"> ■ If the IModelId link of a record in the amAsset table is modified, the agent propagates this change to the IModelId field of the record in the corresponding amPortfolio table. ■ If the IModelId link of a record in the amPortfolio table is modified, the agent propagates this change to the IModelId link of the record in the corresponding amAsset table. 	<ul style="list-style-type: none"> ◆ IModelId

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
FinContAst	<ul style="list-style-type: none">■ PreUpdate on object: IAcquCntrlId■ PreUpdate on object: IReturnEnvId■ PreUpdate on object: dAccept		<ul style="list-style-type: none">■ dEndAcqu■ dStartAcqu■ ILessord■ seAcquMethod■ seAcquStatus

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
		<p>The agent performs the following operations:</p> <ul style="list-style-type: none"> ■ In case of modification of an asset's acquisition contract, it propagates the following information from the new contract to the asset: dEndAcqu, dStartAcqu, lLessorId, seAcquMethod. ■ If the acquisition status of the asset is Not defined, On order or Received and an acceptance date is set, then the acquisition status is automatically set to Received. ■ If the asset is assigned to a return envelope, the acquisition status of the asset is set to To be returned. ■ If the asset was To be returned and then removed from a return envelope, its acquisition status is set to Accepted if the acceptance date is populated. Otherwise, the acquisition status is set to 	

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
LeaseSumAgent	<ul style="list-style-type: none"> ■ Insert on object: amAstCntrDesc ■ Insert on object: amAsset ■ PostDelete on object: amAstCntrDesc ■ PostUpdate on object: lAstId ■ PostUpdate on object: lCntrId ■ PostUpdate on object: mMarketVal ■ PostUpdate on object: MarketValCur ■ PostUpdate on object: mIntPay ■ PostUpdate on object: IntPayCur 	<p style="text-align: center;">Not defined.</p> <p>This agent:</p> <ul style="list-style-type: none"> ■ Makes sure that a contract expressed in one currency cannot be linked to assets expressed in another. If this case arises, an error is returned. ■ Updates the following fields in the contract associated with the asset: mMarketVal, mIntPay, mIntPayTax, depending on the assets linked to the contract. ■ If a link between the asset and a contract is deleted, it recalculates the same information (mMarketVal, mIntPay, mIntPayTax) for the asset. 	<ul style="list-style-type: none"> ■ mMarketVal ■ mIntPay ■ mIntPayTax

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
RentAsset	<ul style="list-style-type: none"> ■ Insert on object: amAssetRent ■ Insert on object: amAsset ■ PostUpdate on object: sePeriodicity ■ PostUpdate on object: bMainRent ■ PostUpdate on object: mPayments ■ PostUpdate on object: mPayments ■ PostUpdate on object: sePeriodicity 	This agent updates the Periodicity and the asset rents associated with the asset using the same information stored at the asset level and vice versa. In addition, it creates an asset rent if this is not already the case.	None in the amAsset table.



5 Assets Included in Projects table (amAstProjDesc)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Assets Included in Projects table. Each section deals with a different type of automatic mechanism.

 Note:

There are no automatic mechanisms other than the default script values on this table.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 5.1. Default value scripts

Object concerned	Script	Description
dIncluded	<code>RetVal = AmDate()</code>	By default, the inclusion date of the asset in the project is the current date.
dRemoved	<code>RetVal = [Project.dEnd]</code>	By default, the removal date of the asset from the project is the project end date.
sSequenceNumber	<pre>If [lAstProjDescId] = 0 Then RetVal = 1 Else RetVal = AmDbGetLong("SELECT ISNULL(MAX(sSequenceNumber), 0)+1 FROM amAstProjDesc where lAstProjDescId =" & [lAstProjDescId]) End If</pre>	By default, the sequence number of the first asset added to the project is set to 1. Otherwise, the sequence number of the asset is the last sequence number incremented by 1.



6 Brands table (amBrand)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Brands table. Each section deals with a different type of automatic mechanism.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 6.1. Default value scripts

Object concerned	Script	Description
Barcode	<code>RetVal = "B" + AmCounter("amBrand_BarCode", 6)</code>	By default, the barcode associated with a brand is the concatenation of the string "B" and the value of the amBrand_BarCode counter on 6 figures.

Table 6.2. Mandatory scripts

Object concerned	Script	Description
Barcode	<code>RetVal = (0<>[bInvent])</code>	If the brand is defined as to be inventoried at barcode inventories (bInvent field set to 1), the Barcode field becomes mandatory.

Integrity rules

There are no integrity rules on the Brands table (**amBrand**).

Agents

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
FullName agent	<ul style="list-style-type: none"> ■ Insert in the am-Brand table ■ Post-Update on the Name field ■ Post-Update on the IParentId link ■ Pre-Update on Name field ■ Pre-Update on IParentId link 	<p>This agent manages tree structures in hierarchic tables.</p> <p>In the Brands table, it maintains hierarchical integrity in the case of sub-brands. The full name of the brand and its hierarchical level are recalculated if:</p> <ul style="list-style-type: none"> ■ A brand is created ■ The name of the brand is modified ■ The parent brand is modified 	<ul style="list-style-type: none"> ■ FullName ■ sLvl



7 | Catalogs table (amCatalog)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Catalogs table. Each section deals with a different type of automatic mechanism.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 7.1. Default value scripts

Object concerned	Script	Description
bExternal	<code>RetVal = 0</code>	By default, the catalog is not accessible externally.
Code	<code>RetVal = AmCounter("amCatalog_Code", 6)</code>	By default, the internal catalog code takes the value of the amCatalog_Code counter on 6 figures
Description	<code>RetVal = [Name]</code>	By default, the description inherits the name of the catalog.
IDefCatSuppld	<code>RetVal = [Contract.lCpyId]</code>	The default supplier of a catalog is the company with which the associated contract was signed.
lLocald	<code>RetVal = [DefSuppCat.lMain Site]</code>	The default location of the catalog is the main location of the default supplier.

Table 7.2. Irrelevance scripts


Object concerned	Script	Description
IDefCatSuppld	<pre>RetVal = 1 if amDbGetLong("SELECT COUNT(Distributors.lCpyId) FROM amCatalog where lCatalogId = " & [lCatalogId]) > 0 then Retval = 0 end if</pre>	This link is only relevant if there are distributor companies for the catalog.
DefSuppCat	<pre>RetVal = 1 if amDbGetLong("SELECT COUNT(Distributors.lCpyId) FROM amCatalog where lCatalogId = " & [lCatalogId]) > 0 then Retval = 0 end if</pre>	This field is only relevant in the case of a department or if the user has administration rights.

Integrity rules

There are no integrity rules on the Catalogs table (**amCatalog**).

Agents

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CCatalogDefSupplier	<ul style="list-style-type: none"> ■ Post-Update on link IDefCatSuppld ■ Pre-Delete on table amRelCatalogSuppliers 	Makes sure the default catalog supplier is in the list of catalog suppliers. If this is not the case, the supplier is added on the fly to this list.	<ul style="list-style-type: none"> ◆ amRelCatalogSuppliers



8 Products table (amCatProduct)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Products table. Each section deals with a different type of automatic mechanism.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 8.1. Validity scripts on the table

Script	Description
<pre>' Check that we have 1 default option per group RetVal = TRUE if [lProdOptId] <> 0 and [bDefaultOption] <> 0 Then if amDbGetLong("SELECT COUNT(1) FROM amCatProduct WHERE lCatProductId <> " & [lCatProductId] & " AND lParentId = " & [lParentId] & " AND OptionGroup.lProdOptId = " & [lProdOptId] & " AND bDefaultOption <> 0 ") <> 0 then Err.Raise(-2009, "There must be one and only one default option per option group for a product.") RetVal = FALSE end if end if</pre>	<p>If there is more than one default option per option group for the product, the record is rejected.</p>

Table 8.2. Default value scripts

Object concerned	Script	Description
bIsPackaged	RetVal = 0	By default, the product is not packaged. Orders for such products are not expressed in packaged units.
bPreinstalled	RetVal = 0 if [lParentId]<>0 then RetVal = 1 end if	If the product has a parent then, by default, it is pre-installed on the product it is a component of.
Certification	RetVal = [Model.Certification]	By default, the certification associated with the product is inherited from the model of the product.
dCertification	RetVal = [Model.dCertification]	By default, the certification date associated with the product is inherited from the model of the product.

Object concerned	Script	Description
Description	<code>RetVal = [Model.Name]</code>	By default, the description of the takes the value of the name of the associated model.
dtPriceCv	<code>RetVal = AmDate()</code>	By default, the conversion date of the average price for the product corresponds to the date of creation of the record.
fPkgQty	<code>RetVal = 1</code>	By default, the quantity per item (expressed in the purchase unit) is set to 1.
fUnitConv	<pre> If [lModelId] <> 0 And [PurchUnit.Dimension] <> [Model.UseUnit.Dimension] Then RetVal = 0 ElseIf [lPurchUnitId] = 0 Then RetVal = 1 Else RetVal = [PurchUnit.fConv] End If </pre>	<ul style="list-style-type: none"> ◆ If the product has an associated model and the dimension (mass, temperature, etc.) in which its unit is expressed is different from that expressed at the model level, then the conversion coefficient for the purchase unit to the unit in which the model is used is zero. ■ If no unit of measurement or packaging is defined for the product, then the coefficient is set to 1. ■ In the other cases, this coefficient is identical to the conversion coefficient defined for the unit of measurement or packaging.
InternalRef	<code>RetVal = AmCounter("InternalRef", 6)</code>	By default, the internal reference of the product takes the value of the InternalRef counter, truncated to 6 figures.
lBrandId	<code>RetVal = [Model.lBrandId]</code>	By default, the brand of the product is inherited from the associated model.
lIconId	<code>RetVal = [Model.lIconId]</code>	By default, the icon associated with the product is that of its associated model.

Object concerned	Script	Description
lPurchUnitId	<code>RetVal = [Model.lUseUnitId]</code>	By default, the unit of measurement or packaging of the product is inherited from the associated model.
lSetQty	<code>RetVal = 1</code>	By default, the number of items in the product packaging is 1.
PriceCur	<code>RetVal = AmDefaultCurrency()</code>	By default, the currency in which the average price of the product is expressed, is the default currency.

Table 8.3. Irrelevance scripts

Object concerned	Script	Description
bDefaultOption	<code>RetVal = (0=[lParentId] or 0=[bOption])</code>	This field, which designates the default option, is only relevant if the product has a parent and the product is an option of its parent product.
bOption	<code>RetVal = (0=[lParentId])</code>	This field is irrelevant if the product does not have a parent.
bPreinstalled	<code>RetVal = (0=[lParentId])</code>	This field is irrelevant if the product does not have a parent.
fPkgQty	<code>'Package Qty is not relevant if lSetQty is irrelevant or fUnitConv is irrelevant RetVal = (0=[lPurchUnitId]) OR (0=[bIsPackaged]) OR (0=[lSetQty])</code>	This field is only relevant in the following cases: <ul style="list-style-type: none"> ■ A unit of measurement or packaging is defined for the product ■ The product is packaged ■ The number of items in the packaged product is not zero
fUnitConv	<code>RetVal = (0=[lPurchUnitId] OR amEvalScript("Irrelevant", "PurchUnit", "")=TRUE)</code>	This field is only relevant if a unit of measurement or packaging is defined for the product

Object concerned	Script	Description
IProdOptId	RetVal = (0=[lParentId] or 0=[bOption])	This field, which designates the default option, is only relevant if the product has a parent and the product is an option of its parent product.
ISetQty	RetVal = (0=[bIsPackaged])	This field is only relevant if the product is packaged.
OptionGroup	RetVal = (0=[lParentId] or 0=[bOption])	This field, which designates the default option, is only relevant if the product has a parent and the product is an option of its parent product.

Agents

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
FullName agent	<ul style="list-style-type: none"> ■ Insert on object: amCatProduct ■ PostUpdate on object: InternalRef ■ PostUpdate on object: IParentId ■ PreUpdate on object: InternalRef ■ PreUpdate on object: IParentId 	<p>This agent manages tree structures in hierarchic tables.</p> <p>In the Products table, it maintains the integrity of the hierarchical structure. The full name of the product and its hierarchical level are recalculated if:</p> <ul style="list-style-type: none"> ■ the internal reference of the product is modified ■ its parent is modified 	<ul style="list-style-type: none"> ■ FullName ■ sLvl



9 | Catalog References table (amCatRef)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Catalog References table. Each section deals with a different type of automatic mechanism.

 Note:

There are no automatic mechanisms other than the default script values on this table.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 9.1. Default value scripts

Object concerned	Script	Description
Certification	<code>RetVal = [CatProduct.Certification]</code>	By default, the certification of a catalog reference is inherited from the product.
Description	<code>RetVal = [CatProduct.Description]</code>	By default, the description of a catalog reference is inherited from the product.
dPriceUpdate	<code>RetVal = AmDate()</code>	By default, the price update date of the catalog reference is the date of creation of the catalog reference.
dtEndValidity	<code>RetVal = [Catalog.dtEndValidity]</code>	By default, the end date of validity of the reference is that of the catalog containing the reference.
dtStartValidity	<code>RetVal = [Catalog.dtStartValidity]</code>	By default, the validity start date of the reference is that of the catalog containing the reference.
fMinQty	<code>RetVal = 1</code>	By default, the minimum orderable quantity for the reference is set to 1.
fPrice	<code>If [CatProduct.PriceCur] = [Catalog.Currency.Name] Then RetVal = [CatProduct.mPrice] Else RetVal = 0 End If</code>	<ul style="list-style-type: none"> ■ If the currency used for the product is identical to that used for the catalog then the purchase price in the catalog reference is inherited from the product. ■ Otherwise, the purchase price is set to 0.

Object concerned	Script	Description
Ref	<code>RetVal = [CatProduct.Description] + " (" + [Catalog.Name] + ")"</code>	By default, the catalog reference number corresponds to the product description.

Table 9.2. Mandatory scripts

Object concerned	Script	Description
IClassCodeId	<code>RetVal = ("<>[Catalog.ProductClass])"</code>	This field, which designates the classification codes, is mandatory if the product has a classification used as a reference.

Integrity rules

There are no integrity rules on the Catalog References table (**amCatRef**).

Agents

The following table lists the active agents on the Catalog References table (**amCatRef**).

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CDateAlarmAgent	<ul style="list-style-type: none"> Post-Update on the dtEndValidity field 	This agent recalculates alarms associated with the End of Validity date of the catalog reference.	None in the amCatRef table.



10 Companies table (amCompany)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Companies table. Each section deals with a different type of automatic mechanism.

 **Note:**

There are no automatic mechanisms other than the scripts on this table.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 10.1. Default value scripts

Object concerned	Script	Description
Code	<code>RetVal = "S" + AmCounter("amCompany_Code", 6)</code>	By default, the unique code associated with the company is the concatenation of the letter C and the value of the amCompany_Code counter on 6 figures.

Table 10.2. Irrelevance scripts

Object concerned	Script	Description
CardTypesAccepted	<code>RetVal = (1=[sePayment])</code>	This link, which points to the card types accepted by the company, is irrelevant if the company does not accept payment cards.
Contacts	<code>RetVal = (0=[lCpyId])</code>	This link, which points to the contracts defined for the company, is irrelevant if the identifier of the company is zero.



11 | Computers table (amComputer)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Computers table. Each section deals with a different type of automatic mechanism.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 11.1. Default value scripts

SQL name of the object concerned	Script	Description
CPUPType	<code>RetVal = [Portfolio.Model.lCPUPType]</code>	By default, this field, which contains the processor type of the computer, inherits the same value as that of the portfolio item model from which it is derived.
ICPUSpeedMHz	<code>RetVal = [Portfolio.Model.lCPUSpeedMHz]</code>	By default, this field, which contains the processor speed of the computer, inherits the same value as that of the portfolio item model from which it is derived.
IDiskSizeMb	<code>RetVal = [Portfolio.Model.lDiskSizeMb]</code>	By default, this field, which contains the hard disk size of the computer, inherits the same value as that of the portfolio item model from which it is derived.
lIconId	<code>RetVal = [Portfolio.lIconId]</code>	By default, this field, which contains the identifier of the icon used to represent the computer, inherits the same value as that of the model from which it is derived.
IMemorySizeMb	<code>RetVal = [Portfolio.Model.lMemorySizeMb]</code>	By default, this field, which contains the RAM size of the computer, inherits the same value as that of the portfolio item model from which it is derived.

SQL name of the object concerned	Script	Description
Name	<pre> if [bGroup]=0 then RetVal = "CPU" + AmCounter ("amComputer_Name", 6) else RetVal = "GRP" + AmCounter ("amComputer_Group", 6) end if </pre>	<p>This script enables you to automatically name a computer or computer group:</p> <ul style="list-style-type: none"> ■ If it is not a computer group (<code>[bGroup] = 0</code>), the name assigned by default is the result of concatenating the "CPU" string and the value of the am-Computer_Name counter on 6 figures. ■ If it is a computer group, the name assigned by default is the result of concatenating the string "GRP" and the value of the am-Computer_Group counter on 6 figures. <p>Note:</p> <p>For further information on the AmCounter(), refer to the AssetCenter Programmer's Reference.</p> <p>For further information on counter, refer to the Administration guide, chapter Standard database description files, section Customizing the database/ Counters in field default values.</p>
TcplpAddress	<pre> if [bGroup] <> 0 then RetVal = "" else RetVal = [Name] end if </pre>	<p>This script is useful for computer groups. In this case, it populates the field that usually contains the IP address of the computer with the name of the computer group. If it is not a group (<code>[bGroup] <> 0</code>), the field is left empty.</p>

SQL name of the object concerned	Script	Description
TcpIpHostName	<pre> if [bGroup] <> 0 then RetVal = "" else RetVal = [Name] end if </pre>	This script is useful for computer groups. In this case, it populates the field that usually contains the IP name of the computer with the name of the computer group. If it is not a group ([bGroup] <> 0), the field is left empty.

Table 11.2. Irrelevance scripts

Object concerned	Script
BIOSAssetTag	RetVal = (0<> [bGroup])
BIOSSource	RetVal = (0<> [bGroup])
bTcpIpRouting	RetVal = (0<> [bGroup])
ComputerDesc	RetVal = (0<> [bGroup])
ComputerType	RetVal = (0<> [bGroup])
CPUInternal	RetVal = (0<> [bGroup])
CPUType	RetVal = (0<> [bGroup])
dtBIOS	RetVal = (0<> [bGroup])
dtHardScan	RetVal = (0<> [bGroup])
dtLastScan	RetVal = (0<> [bGroup])
dtNetworkScan	RetVal = (0<> [bGroup])
dtNextScan	RetVal = (0<> [bGroup])
dtSoftScan	RetVal = (0<> [bGroup])
lpxSpxAddress	RetVal = (0<> [bGroup])
lpxSpxServer	RetVal = (0<> [bGroup])
IColorDepth	RetVal = (0<> [bGroup])
ICPUNumber	RetVal = (0<> [bGroup])
ICPUSpeedMHz	RetVal = (0<> [bGroup])
IDiskSizeMb	RetVal = (0<> [bGroup])
IHorizontalRes	RetVal = (0<> [bGroup])
IItemId	RetVal = (0<> [bGroup])
IMemorySizeMb	RetVal = (0<> [bGroup])
IScanHistId	RetVal = (0<> [bGroup])
ISwapSizeMb	RetVal = (0<> [bGroup])
IVerticalRes	RetVal = (0<> [bGroup])
OperatingSystem	RetVal = (0<> [bGroup])
OSBuildNumber	RetVal = (0<> [bGroup])
OSDirectory	RetVal = (0<> [bGroup])

Object concerned	Script
OSLocale	RetVal = (0<> [bGroup])
OSServiceLevel	RetVal = (0<> [bGroup])
PhysicalAddress	RetVal = (0<> [bGroup])
ScannerDesc	RetVal = (0<> [bGroup])
ScannerVersion	RetVal = (0<> [bGroup])
SoundCard	RetVal = (0<> [bGroup])
TcplpAddress	RetVal = (0<> [bGroup])
TcplpDomain	RetVal = (0<> [bGroup])
TcplpHostName	RetVal = (0<> [bGroup])
VideoCard	RetVal = (0<> [bGroup])
Workgroup	RetVal = (0<> [bGroup])
Agents	RetVal = (0<> [bGroup])
DaTracking	RetVal=(0<> [bGroup])
ExtensionCards	RetVal = (0<> [bGroup])
LogicalDrives	RetVal = (0<> [bGroup])
NetworkCards	RetVal = (0<> [bGroup])
PhysicalDrives	RetVal = (0<> [bGroup])
Portfolio	RetVal = (0<> [bGroup])
ScanHistory	RetVal = (0<> [bGroup])
SubGroups	RetVal = (0= [bGroup])

The following objects share the same irrelevance script:

```
RetVal = (0= [bGroup] )
```

In the case of a computer group, they are not relevant. There are therefore not displayed.

Integrity rules

There are no integrity rules on the Computers table (**amComputer**).

Agents

The following table lists the agents working on the Computers table (**amComputer**).

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CGbAssetAssignment	<ul style="list-style-type: none"> ■ Insert in amComputer table. ■ Pre-Commit of a transaction when it impacts the amComputer table. 	<p>The Computers table is an overflow table of the Portfolio Items table.</p> <p>When a record is created in the amComputer table, a record is created in the reference table - in this case the Portfolio Items table (amPortfolio) - except if the overflow link is irrelevant, which is the case for computer groups. A record is also created in the Assets table (amAsset).</p> <p>Note:</p> <p>For further information on overflow tables, refer to the Portfolio guide, chapter Overview (Portfolio), section Overflow tables.</p>	None in the amComputer table.

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CRedundancyAgent	<ul style="list-style-type: none"> ■ Insert in the amComputer table ■ Insert in the amPortfolio table ■ Post-Update on the ItemId link in the amComputer table. ■ Post-Update on the AssetTag field in the amComputer table. ■ Post-Update on the AssetTag field in the amPortfolio table. 	<p>This agent makes sure that the AssetTag fields of a computer and its associated portfolio item are identical:</p> <ul style="list-style-type: none"> ■ If the AssetTag field of a record in the amComputer table is modified, the agent propagates this change to the AssetTag field of the record in the corresponding amPortfolio table. ■ If the AssetTag field of a record in the amPortfolio table is modified, the agent propagates this change to the AssetTag field of the record in the corresponding amComputer table. 	<ul style="list-style-type: none"> ■ AssetTag field in the amComputer table. ■ AssetTag field in the amPortfolio table.

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
FullName	<ul style="list-style-type: none"> ■ Insert in the am-Computer table ■ Post-Update on the Name field ■ Post-Update on the bGroup field ■ Post-Update on the IParentId link ■ Pre-Update on the Name field ■ Pre-Update on the bGroup field ■ Pre-Update on the IParentId link 	<p>This agent manages tree structures in hierarchic tables.</p> <p>In the Computers table, it maintains hierarchical integrity in the case of computer groups. The full name of the computer and its hierarchical level are recalculated if:</p> <ul style="list-style-type: none"> ■ the name of the computer or its parent group is modified ■ the group of the computer (of its parent) is modified ■ the computer is changed to a computer group or vice-versa 	<ul style="list-style-type: none"> ■ FullName field ■ sLvl field

Workflows

The following tables summarize the workflows dealing with the Computers table (**amComputer**).

Warning:

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Determining the workflows used for a table](#) [page 183] at the end of this document.

Workflow reference	Workflow type	Description
BST_SAM20	Synchronous	This workflow updates the installed software not detected at the last scan by setting their Assignment (seAssignment) field to Missing . It is automatically triggered when the Last software inventory (dtSoftScan) field is updated.
STD_PROCUR_POPULATED	Synchronous	This workflow updates the information on a computer. It is triggered automatically when a record is created in the am-Computer table.
DA_SCAN	Asynchronous	This workflow, triggered at regular intervals, starts the computer scan according to the options specified in the database.



12 | Contacts table (amContact)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Contracts table. Each section deals with a different type of automatic mechanism.

 **Note:**

There are no automatic mechanisms other than the default script values on this table.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 12.1. Default value scripts

Object concerned	Script	Description
Fax	<code>RetVal = [Company.Fax]</code>	By default, the fax number of a contact is that of their company.
Phone	<code>RetVal = [Company.Phone]</code>	By default, the telephone number of a contact is that of their company.



13 | Contracts table (amContract)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Contracts table. Each section deals with a different type of automatic mechanism.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 13.1. Validity scripts on the table

Script	Description
<pre>If Not IsEmpty([dEnd]) and Not IsEmpty([dStart]) and [dStart] > [dEnd] Then Err.Raise(-2009, "The end date (dEnd) must be greater than or equal to the start date (dStart).") RetVal = FALSE Else RetVal = TRUE End If</pre>	<p>If the start and end dates of the contract are not empty and the end date comes before the start date, the record is rejected.</p>

Table 13.2. Default value scripts

Object concerned	Script	Description
AmountCur	RetVal = AmDefaultCurrency ()	By default, the amount of the contract is expressed in the default currency.
AssignCond	RetVal = [Parent.AssignCond]	By default, the assignment conditions of a contract are inherited from its parent contract.
AstIntPayTaxCur	RetVal = AmDefaultCurrency ()	By default, the tax on the interims of the assets on the contract are expressed in the default currency.
bAssignable	RetVal = [Parent.bAssignable]	By default, the possibility of assigning a contract is inherited from its parent.
bPurchOpt	RetVal = [Parent.bPurchOpt]	By default, the possibility of purchasing the assets on a contract is inherited from its parent.
bRenOpt	RetVal = [Parent.bRenOpt]	By default, the possibility of renewing the assets on a contract is inherited from its parent.
bRetOpt	RetVal = [Parent.bRetOpt]	By default, the possibility of returning the assets on a contract is inherited from its parent.

Object concerned	Script	Description
bUpgOpt	<code>RetVal = [Parent.bUpgOpt]</code>	By default, the possibility of upgrading the contract is inherited from its parent.
dEnd	<code>If [lParentId]<>0 OR [Model.tsCntrDuration]=0 Then RetVal = [Parent.dEnd] Else RetVal = AmDateAddLogical([dStart], [Model.tsCntrDuration]) End If</code>	If the contract has a parent contract, or the planned duration of the contracts at the model level is zero, then the default end date of the contract is that of the parent contract. Otherwise, the contract end date is calculated by adding the length specified at the model level to the contract start date.
dPurchNotice	<code>RetVal = [Parent.dPurchNotice]</code>	By default, the purchase notice date for the contract is inherited from its parent.
dRenNotice	<code>RetVal = [Parent.dRenNotice]</code>	By default, the renewal notice date for the contract is inherited from its parent.
dRetNotice	<code>RetVal = [Parent.dRetNotice]</code>	By default, the return notice date for the contract is inherited from its parent.
dStart	<code>If [lParentId]<>0 Then RetVal = [Parent.dStart] Else RetVal = Date() End If</code>	If the contract has a parent contract, the contract start date is inherited from the parent. Otherwise, it is contract creation date.
dtAmountCv	<code>RetVal = AmDate()</code>	By default, the conversion date for the amount of the contract is the creation date of the record.
dtAstIntPayTaxCv	<code>RetVal = AmDate()</code>	By default, the conversion date for the tax on the interim rent for the assets on the contract is the record creation date.
dtIntPayAstCv	<code>RetVal = AmDate()</code>	By default, the conversion date for the interim rent for the assets on the contract is the record creation date.

Object concerned	Script	Description
dtIntPayCv	<code>RetVal = AmDate()</code>	By default, the conversion date for the interim rent of the contract is the record creation date.
dtIntPayTaxCv	<code>RetVal = AmDate()</code>	By default, the conversion date for the tax on the interim rent of the contract is the record creation date.
dtMarketValCv	<code>RetVal = AmDate()</code>	By default, the conversion date for the total value of the assets on the contract is the record creation date.
dtPOCommitmentCv	<code>RetVal = AmDate()</code>	By default, the conversion date for the contract commitment is the creation date of the record.
IntPayAstCur	<code>RetVal = AmDefaultCurrency()</code>	By default, the interim rents of the assets on the contract are expressed in the default currency.
IntPayCur	<code>RetVal = AmDefaultCurrency()</code>	By default, the interim rent for the contract is expressed in the default currency.
IntPayTaxCur	<code>RetVal = AmDefaultCurrency()</code>	By default, the tax on the interim rent for the contract is expressed in the default currency.
lAssigneeld	<code>RetVal = [Parent.lAssigneeId]</code>	By default, the contact assignee is inherited from its parent.
lBillAddrId	<code>RetVal = [Parent.lBillAddrId]</code>	By default, the billing address of a contract is inherited from its parent.
lBillCnctId	<code>RetVal = [Parent.lBillCnctId]</code>	By default, the billing contact of a contract is inherited from its parent.
lCntrCnctId	<code>RetVal = [Parent.lCntrCnctId]</code>	By default, the contact of a contract is inherited from its parent.
lCostCatId	<code>If [lParentId] <> 0 Then RetVal = [Parent.lCostCatId] Else RetVal = [Model.lCostCatId] End If</code>	If the contract has a parent contract, the cost category of the contract is inherited from its parent. Otherwise, it is inherited from its model.

Object concerned	Script	Description
ICostId	<code>RetVal = [Parent.lCostId]</code>	By default, the cost center of a contract is inherited from its parent.
ICpyId	<code>RetVal = [POrdLine.POrder.lSuppId] If RetVal=0 Then RetVal = [Parent.lCpyId] End If</code>	If the contract has a parent contract, the company associated with the contract is inherited from the parent. Otherwise, the company is taken from the supplier specified in the order line giving rise to the contract.
IIconId	<code>RetVal = [Model.lIconId]</code>	By default, the icon used to represent the contract is identical to that used for the model.
IInsurCnctId	<code>RetVal = [Parent.lInsurCnctId]</code>	By default, the insurance contract of a contract is inherited from its parent.
ILessorId	<code>RetVal = [Parent.lLessorId]</code>	By default, the lessor associated with a contract is inherited from its parent.
ILossValRuleId	<code>RetVal = [Parent.lLossValRuleId]</code>	By default, the loss value rule associated with a contract is inherited from its parent.
INotifAddrId	<code>RetVal = [Parent.lNotifAddrId]</code>	By default, the notification address of a contract is inherited from its parent.
LossCond	<code>RetVal = [Parent.LossCond]</code>	By default, the lessor indemnification conditions in case of loss or destruction of assets are inherited from the parent.
ISupervId	<code>RetVal = [Parent.lSupervId]</code>	By default, the contact supervisor is inherited from its parent.
ITechCnctId	<code>RetVal = [Parent.lTechCnctId]</code>	By default, the technical contact of a contract is inherited from its parent.
MarketValCur	<code>RetVal = AmDefaultCurrency()</code>	By default, the total value of the assets on the contract is expressed in the default currency.

Object concerned	Script	Description
Nature	<pre>If [Parent] <> 0 Then RetVal = [Parent.Nature] End If</pre>	By default, the nature of a contract is inherited from its parent.
pDefLRF	<pre>RetVal = [Parent.pDefLRF]</pre>	By default, the lease rate factor for a contract is inherited from its parent.
pDefRenPercent	<pre>RetVal = [Parent.pDefRenPercent]</pre>	By default, the percentage to apply to the previous rent to determine the renewed rent payments, is inherited from the parent contract.
pIntRentPercent	<pre>RetVal = [Parent.pIntRentPercent]</pre>	By default, the percentage associated with a contract is inherited from its parent.
POCommitmentCur	<pre>RetVal = AmDefaultCurrency()</pre>	By default, the commitment amount associated with the contract is expressed in the default currency.
PurchOptType	<pre>RetVal = [Parent.PurchOptType]</pre>	By default, the purchase option type of a contract is inherited from its parent.
Purpose	<pre>RetVal = [Model.Name]</pre>	By default, the contract purpose takes the value of the name of the model associated with the contract.
Ref	<pre>RetVal = "C" + AmCounter("amContract_Ref", 6)</pre>	By default, the contract reference is the concatenation of the letter C and the value of the amContract_Ref counter on 6 figures.
RenOptType	<pre>RetVal = [Parent.RenOptType]</pre>	By default, the renewal option type of a contract is inherited from its parent.
RetOptType	<pre>RetVal = [Parent.RetOptType]</pre>	By default, the return option type of a contract is inherited from its parent.
seAcquMethod	<pre>RetVal = 2</pre>	By default, the acquisition method for the assets on the contract is Lease .
seFreightOutPayer	<pre>RetVal = [Parent.seFreightOutPayer]</pre>	By default, whether freight out costs are payable by the lessor or the lessee is inherited from the parent contract.

Object concerned	Script	Description
seInstallCountType	if [lModelId] > 0 Then RetVal = [Model.seSoftLicType]	By default, software installations count type is inherited from the license type defined at the model level.
seInsurPayer	RetVal = [Parent.seInsurPayer]	By default, whether the insurance costs are payable by the lessor or by the lessee is inherited from the parent contract.
seIntRentType	RetVal = [Parent.seIntRentType]	By default, the calculation method for interim rent is inherited from the parent contract.
seLossValCalcMode	RetVal = [Parent.seLossValCalcMode]	By default, the calculation method for loss values is inherited from the parent contract.
sePayType	RetVal = [Parent.sePayType]	By default, the nature of payments of a contract is inherited from its parent.
sePeriodicity	RetVal = 360	By default, the frequency of payment for contract rents is annual.
sePlannedOpt	RetVal = [Parent.sePlannedOpt]	By default, the planned end-of-contract option is inherited from the parent contract.
seShipCostPayer	RetVal = [Parent.seShipCostPayer]	By default, whether the shipping costs are payable by the lessor or by the lessee is inherited from the parent contract.
seStatus	RetVal = 0	By default, the contract is In preparation .
seType	<pre>If [lModelId] <> 0 Then RetVal = [Model.seContractType] ElseIf [Parent.seType] = 1 Then RetVal = 2 Else RetVal = 0 End If</pre>	<ul style="list-style-type: none"> ■ If there is a model associated with the contract, then the contract type is derived from that specified at the model level. ■ If no model is associated with the contract and the parent contract type is Master lease, then, by default, the contract is Lease schedule. ■ In the other cases, the contract type is Other.

Object concerned	Script	Description
Status	RetVal = [Parent.Status]	By default, the status of a contract is inherited from its parent.
tsDefRenDur	RetVal = [Parent.tsDefRenDur]	By default, the renewal period of a contract is inherited from its parent.
tsLessorNotice	RetVal = [Parent.tsLessorNotice]	By default, the notification period for any modifications to the contract is inherited from the parent contract.
tsNotice	RetVal = [Parent.tsNotice]	By default, the notice period of a contract is inherited from its parent.
tsPurchNotice	RetVal = [Parent.tsPurchNotice]	By default, the minimum purchase notice period for assets before the end of a contract is inherited from the parent contract.
tsRenNotice	RetVal = [Parent.tsRenNotice]	By default, the minimum renewal notice period for assets before the end of a contract is inherited from the parent contract.
tsRetNotice	RetVal = [Parent.tsRetNotice]	By default, the minimum return notice period for assets before the end of a contract is inherited from the parent contract.
UpgOptType	RetVal = [Parent.UpgOptType]	By default, the upgrade option type is inherited from the parent contract.

Table 13.3. Read-Only scripts

Object concerned	Script	Description
seType	<pre>If [Parent.seType] = 1 Then RetVal = 1 Else RetVal = 0 End If</pre>	If the parent contract is a Master lease , then the field containing the contract Type is read only.

Table 13.4. Irrelevance scripts

Object concerned	Script	Description
AssignCond	<pre>RetVal = 0 if [seType]<>1 and [seType] <>2 then RetVal = 1 end if</pre>	This field is only relevant if the contract is a Master lease or Lease schedule .
bAssignable	<pre>RetVal = 0 if [seType]<>1 and [seType] <>2 then RetVal = 1 end if</pre>	This field is only relevant if the contract is a Master lease or Lease schedule .
bPurchOpt	<pre>RetVal = ([seType]<>1 and [seType]<>2)</pre>	This field is only relevant if the contract is a Master lease or Lease schedule .
bRenOpt	<pre>RetVal = ([seType]<>1 and [seType]<>2)</pre>	This field is only relevant if the contract is a Master lease or Lease schedule .
bRetOpt	<pre>RetVal = ([seType]<>1 and [seType]<>2)</pre>	This field is only relevant if the contract is a Master lease or Lease schedule .
bUpgOpt	<pre>RetVal = ([seType]<>1 and [seType]<>2)</pre>	This field is only relevant if the contract is a Master lease or Lease schedule .
dPurchNotice	<pre>if amEvalScript("Irrelevan t", "PurchOptType", "")<>0 OR [seType]<>2 then RetVal = 1 else RetVal = 0 end if</pre>	This field is irrelevant if the Purchase option type is irrelevant or if the contract is not a Lease schedule .
dRenNotice	<pre>if amEvalScript("Irrelevan t", "RenOptType", "")<>0 O R [seType]<>2 then RetVal = 1 else RetVal = 0 end if</pre>	This field is irrelevant if the Renewal option type is irrelevant or if the contract is not a Lease schedule .
dRetNotice	<pre>if amEvalScript("Irrelevan t", "RetOptType", "")<>0 O R [seType]<>2 then RetVal = 1 else RetVal = 0 end if</pre>	This field is irrelevant if the Return option type is irrelevant or if the contract is not a Lease schedule .
lAssigneeld	<pre>RetVal = 0 if [seType]<>1 and [seType] <>2 then RetVal = 1 end if</pre>	This field is only relevant if the contract is a Master lease or Lease schedule .

Object concerned	Script	Description
ICpyld	<pre>RetVal = (amEvalScript("Irrelevant", "Lessor", "")=FALSE)</pre>	This field is only relevant if the Lessor link is irrelevant.
IDefPOrdId	<pre>RetVal = [seType] <>6</pre>	This field is only relevant if the contract Type is Blanket PO .
ILessorId	<pre>RetVal = 0 if [seType] <>1 and [seType] <>2 then RetVal = 1 end if</pre>	This field is only relevant if the contract is a Master lease or Lease schedule .
ILossValRuleId	<pre>RetVal = 0 if [seType] <>1 and [seType] <>2 then RetVal = 1 end if</pre>	This field is only relevant if the contract is a Master lease or Lease schedule .
IOptCmtId	<pre>RetVal = ([seType] <>1 and [seType] <>2)</pre>	This field is only relevant if the contract is a Master lease or Lease schedule .
LossCond	<pre>RetVal = 0 if [seType] <>1 and [seType] <>2 then RetVal = 1 end if</pre>	This field is only relevant if the contract is a Master lease or Lease schedule .
IPurchOptCmtId	<pre>RetVal = 0 if amEvalScript("Irrelevant", "bPurchOpt", "") <>0 or [bPurchOpt]=0 then RetVal = 1 end if</pre>	This field is irrelevant if the Purchase field is irrelevant or the assets on the contract cannot be purchased (bPurchOpt=0).
IRenOptCmtId	<pre>RetVal = 0 if amEvalScript("Irrelevant", "bRenOpt", "") <>0 or [bRenOpt]=0 then RetVal = 1 end if</pre>	This field is irrelevant if the Renewal field is irrelevant or the assets on the contract cannot be renewed (bRenOpt=0).
IRetOptCmtId	<pre>RetVal = 0 if amEvalScript("Irrelevant", "bRetOpt", "") <>0 or [bRetOpt]=0 then RetVal = 1 end if</pre>	This field is irrelevant if the Return field is irrelevant or the assets on the contract cannot be returned (bRetOpt=0).
IUpgOptCmtId	<pre>RetVal = 0 if amEvalScript("Irrelevant", "bUpgOpt", "") <>0 or [bUpgOpt]=0 then RetVal = 1 end if</pre>	This field is irrelevant if the Upgrade field is irrelevant or the assets on the contract cannot be upgraded (bUpgOpt=0).
mMarketVal	<pre>RetVal = [seType] <>2</pre>	This field is only relevant if the contract Type is Lease schedule .

Object concerned	Script	Description
mPOCommitment	RetVal = [seType]<>6	This field is only relevant if the contract Type is Blanket PO .
pDefLRF	RetVal = 0 if [seType]<>1 and [seType] <>2 then RetVal = 1 end if	This field is only relevant if the contract is a Master lease or Lease schedule .
pDefRenPercent	RetVal = 0 if amEvalScript("Irrelevant", "bRenOpt", "")<>0 or [bRenOpt]=0 then RetVal = 1 end if	This field is irrelevant if the Renewal field is irrelevant or the assets on the contract cannot be renewed (bRenOpt=0).
pIntRentPercent	RetVal = [seType]<>1	This field is only relevant if the contract Type is Master lease .
PurchOptType	RetVal = 0 if amEvalScript("Irrelevant", "bPurchOpt", "")<>0 or [bPurchOpt]=0 then RetVal = 1 end if	This field is irrelevant if the Purchase field is irrelevant or the assets on the contract cannot be purchased (bPurchOpt=0).
RenOptType	RetVal = 0 if amEvalScript("Irrelevant", "bRenOpt", "")<>0 or [bRenOpt]=0 then RetVal = 1 end if	This field is irrelevant if the Renewal field is irrelevant or the assets on the contract cannot be renewed (bRenOpt=0).
RetOptType	RetVal = 0 if amEvalScript("Irrelevant", "bRetOpt", "")<>0 or [bRetOpt]=0 then RetVal = 1 end if	This field is irrelevant if the Return field is irrelevant or the assets on the contract cannot be returned (bRetOpt=0).
seAcquMethod	RetVal = ([seType]<>1 and [seType]<>2)	This field is only relevant if the contract is a Master lease or Lease schedule .
seFreightOutPayer	RetVal = 0 if [seType]<>1 and [seType] <>2 then RetVal = 1 end if	This field is only relevant if the contract is a Master lease or Lease schedule .
seInsurPayer	RetVal = 0 if [seType]<>1 and [seType] <>2 then RetVal = 1 end if	This field is only relevant if the contract is a Master lease or Lease schedule .
seIntRentType	RetVal = 0 if [seType]<>1 and [seType] <>2 then RetVal = 1 end if	This field is only relevant if the contract is a Master lease or Lease schedule .

Object concerned	Script	Description
seLossValCalcMode	RetVal = 0 if [seType]<>1 and [seType] >>2 then RetVal = 1 end if	This field is only relevant if the contract is a Master lease or Lease schedule .
sePlannedOpt	RetVal = ([seType]<>1 and [seType]<>2)	This field is only relevant if the contract is a Master lease or Lease schedule .
seShipCostPayer	RetVal = 0 if [seType]<>1 and [seType] >>2 then RetVal = 1 end if	This field is only relevant if the contract is a Master lease or Lease schedule .
tsDefRenDur	RetVal = 0 if amEvalScript("Irrelevant", "bRenOpt", "")<>0 or [bRenOpt]=0 then RetVal = 1 end if	This field is irrelevant if the Renewal field is irrelevant or the assets on the contract cannot be renewed (bRenOpt=0).
tsLessorNotice	RetVal = 0 if [seType]<>1 and [seType] >>2 then RetVal = 1 end if	This field is only relevant if the contract is a Master lease or Lease schedule .
tsPurchNotice	RetVal = 0 if amEvalScript("Irrelevant", "bPurchOpt", "")<>0 or [bPurchOpt]=0 then RetVal = 1 end if	This field is irrelevant if the Purchase field is irrelevant or the assets on the contract cannot be purchased (bPurchOpt=0).
tsRenNotice	RetVal = 0 if amEvalScript("Irrelevant", "bRenOpt", "")<>0 or [bRenOpt]=0 then RetVal = 1 end if	This field is irrelevant if the Renewal field is irrelevant or the assets on the contract cannot be renewed (bRenOpt=0).
tsRetNotice	RetVal = 0 if amEvalScript("Irrelevant", "bRetOpt", "")<>0 or [bRetOpt]=0 then RetVal = 1 end if	This field is irrelevant if the Return field is irrelevant or the assets on the contract cannot be returned (bRetOpt=0).
UpgOptType	RetVal = 0 if amEvalScript("Irrelevant", "bUpgOpt", "")<>0 or [bUpgOpt]=0 then RetVal = 1 end if	This field is irrelevant if the Upgrade field is irrelevant or the assets on the contract cannot be upgraded (bUpgOpt=0).

Object concerned	Script	Description
Assignee	RetVal = 0 if [seType]<>1 and [seType] <>2 then RetVal = 1 end if	This field is only relevant if the contract is a Master lease or Lease schedule .
AstCntrDescs	RetVal = ([seType]=1 or [seType]=2 or [seType]=6)	This field is irrelevant if the contract Type is Master lease, Lease schedule or Blanket PO .
Company	RetVal = (amEvalScript("Irrelevant", "Lessor", "")=FALSE)	This field is only relevant if the Lessor link is irrelevant.
DefPOrder	RetVal = [seType]<>6	This field is only relevant if the contract Type is Blanket PO .
ExpenseLines	RetVal = ([seType]=1)	This field is irrelevant if the contract Type is Master lease .
LeasedAssets	RetVal = [seType]<>2	This field is only relevant if the contract Type is Lease schedule .
Lessor	RetVal = 0 if [seType]<>1 and [seType] <>2 then RetVal = 1 end if	This field is only relevant if the contract is a Master lease or Lease schedule .
Licenses	RetVal = [seType]<>5	This field is only relevant if the contract Type is License .
Loans	RetVal = ([seType]=1 or [sePayType]=-1 or [sePayType]=0)	This field is irrelevant if the contract Type is Master lease , or if the Nature of payments is None or Rents .
LossValRule	RetVal = 0 if [seType]<>1 and [seType] <>2 then RetVal = 1 end if	This field is only relevant if the contract is a Master lease or Lease schedule .
OptCmt	RetVal = ([seType]<>1 and [seType]<>2)	This field is only relevant if the contract is a Master lease or Lease schedule .
POrdersBlanketPO	RetVal = [seType]<>6	This field is only relevant if the contract Type is Blanket PO .
PurchOptCmt	RetVal = 0 if amEvalScript("Irrelevant", "bPurchOpt", "")<>0 or [bPurchOpt]=0 then RetVal = 1 end if	This field is irrelevant if the Purchase field is irrelevant or the assets on the contract cannot be purchased (bPurchOpt=0).

Object concerned	Script	Description
RenOptCmt	RetVal = 0 if amEvalScript("Irrelevant", "bRenOpt", "")<>0 or [bRenOpt]=0 then RetVal = 1 end if	This field is irrelevant if the Renewal field is irrelevant or the assets on the contract cannot be renewed (bRenOpt=0).
Rents	RetVal = ([seType]=1 or [sePayType]=-1 or [sePayType]=1)	This field is irrelevant if the contract Type is Master lease , or if the Nature of payments is None or Rents .
RetOptCmt	RetVal = 0 if amEvalScript("Irrelevant", "bRetOpt", "")<>0 or [bRetOpt]=0 then RetVal = 1 end if	This field is irrelevant if the Return field is irrelevant or the assets on the contract cannot be returned (bRetOpt=0).
Schedules	RetVal = [seType]<>1	This field is only relevant if the contract Type is Master lease .
UpgOptCmt	RetVal = 0 if amEvalScript("Irrelevant", "bUpgOpt", "")<>0 or [bUpgOpt]=0 then RetVal = 1 end if	This field is irrelevant if the Upgrade field is irrelevant or the assets on the contract cannot be upgraded (bUpgOpt=0).
WorkOrders	RetVal = [seType]<>4	This field is only relevant if the contract Type is Maintenance .

Integrity rules

Name of the rule	List of monitored objects	Rule(s) verified	List of any modified objects
CNotifContDateInteg	<ul style="list-style-type: none"> ■ SyncRead on object: dEnd ■ SyncRead on object: dPurchNotice ■ SyncRead on object: tsPurchNotice 	Ensures the consistency between the End date , the Purchase notif. date and the Purchase notice period of the contract. The contract end date is always kept. The last value entered by the user is the kept to the detriment of the remaining value.	<ul style="list-style-type: none"> ■ dPurchNotice ■ tsPurchNotice

Name of the rule	List of monitored objects	Rule(s) verified	List of any modified objects
CNotifContDateInteg	<ul style="list-style-type: none"> ■ SyncRead on object: dEnd ■ SyncRead on object: dRenNotice ■ SyncRead on object: tsRenNotice 	Ensures the consistency between the End date , the Renewal notice date and the Renewal notice period of the contract. The contract end date is always kept. The last value entered by the user is the kept to the detriment of the remaining value.	<ul style="list-style-type: none"> ■ dRenNotice ■ tsRenNotice
CNotifContDateInteg	<ul style="list-style-type: none"> ■ SyncRead on object: dEnd ■ SyncRead on object: dRetNotice ■ SyncRead on object: tsRetNotice 	Ensures the consistency between the End date , the Return notice date and the Return notice period of the contract. The contract end date is always kept. The last value entered by the user is the kept to the detriment of the remaining value.	<ul style="list-style-type: none"> ■ dRetNotice ■ tsRetNotice

Agents

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CContDateInteg	<ul style="list-style-type: none"> ■ SyncRead on object: dStart ■ SyncRead on object: dEnd ■ SyncRead on object: tsDuration 	Ensures the consistency between the start and end dates of the contract and its length. The last value entered by the user is kept to the detriment of the others.	<ul style="list-style-type: none"> ■ dEnd ■ dStart ■ tsDuration

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CContractInherit3rdPartyAgent	<ul style="list-style-type: none"> ◆ Insert on object: amContract 	On creating a contract, the information on third-party companies is copied over from the parent, if there is one.	
CContractLink3rdPartyAgent	<ul style="list-style-type: none"> ■ PostUpdate on object: lLessord ■ PostUpdate on object: lAssigneeld 	Makes sure that these two links belong to the list of third-party companies of the contract.	
CDateAlarmAgent	<ul style="list-style-type: none"> ◆ PostUpdate on object: dRenNotice 	This agent, if necessary, recalculates the alarms associated with the contract renewal notification date.	
CDateAlarmAgent	<ul style="list-style-type: none"> ◆ PostUpdate on object: dRetNotice 	This agent, if necessary, recalculates the alarms associated with the contract return notification date.	
CDateAlarmAgent	<ul style="list-style-type: none"> ◆ PostUpdate on object: dEnd 	This agent, if necessary, recalculates the alarms associated with the contract end date.	
CDateAlarmAgent	<ul style="list-style-type: none"> ◆ PostUpdate on object: dPurchNotice 	This agent, if necessary, recalculates the alarms associated with the contract purchase (buyout) notification date.	

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CGbAssetAssignment	<ul style="list-style-type: none"> ■ Insert on object: amCable ■ Insert on object: amContract ■ Insert on object: amComputer ■ Insert on object: amSoftInstall ■ Insert on object: amAsset ■ Insert on object: amPortfolio ■ Insert on object: amTraining ■ Insert on object: amWorkOrder ■ Insert on object: amPhone ■ PostDelete on object: amPortfolio ■ PreUpdate on object: IModelId ■ PreUpdate on object: IModelId ■ PreUpdate on object: IModelId ■ PreUpdate on object: IModelId ■ PreUpdate on object: IModelId 	In case of creation of a portfolio item, if necessary this agent creates the corresponding record in the Assets table and the appropriate record in the overflow table matching the management constraint associated with the model of the portfolio item.	

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CLSLossLineAgent	<ul style="list-style-type: none"> ■ Insert on object: amContract ■ PostDelete on object: amPortfolio ■ PostUpdate on object: dStart ■ PostUpdate on object: dEnd ■ PostUpdate on object: mMarketVal ■ PostUpdate on object: lLossValRuleId 	Applies the loss-value rule according to the monitored fields. The contract loss-value records are created.	
ContCompany	<ul style="list-style-type: none"> ■ PreUpdate on object: seType ■ PreUpdate on object: lCpyId ■ PreUpdate on object: lLessorId 	If the contract Type is neither Master lease , nor Lease schedule , the link to the lessor is deleted. For other contract types, any change to the lessor is carried over to the Company with which the contract is signed.	<ul style="list-style-type: none"> ■ lCpyId ■ lLessorId
ContVerslnit	<ul style="list-style-type: none"> ◆ PreUpdate on object: mIntPayAst 	Propagates modifications to the total of initial payments for assets financed by the contract to the initial payment of the contract.	<ul style="list-style-type: none"> ■ IntPayCur ■ mIntPay
COverflow-ChangeAgent	<ul style="list-style-type: none"> ◆ PreUpdate on object: lModelId 	This agent stops a contract model from being changed if this means changing the associated overflow table.	

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
FullName agent	<ul style="list-style-type: none"> ■ Insert on object: amContract ■ PostUpdate on object: Ref ■ PostUpdate on object: IParentId ■ PreUpdate on object: Ref ■ PreUpdate on object: IParentId 	<p>This agent manages tree structures in hierarchic tables.</p> <p>In the Contracts table, it maintains the integrity of the hierarchical structure. The full name of the contract and its hierarchical level are recalculated if:</p> <ul style="list-style-type: none"> ■ the contract reference code is modified ■ its parent is modified 	<ul style="list-style-type: none"> ■ FullName ■ sLvl
RentContract	<ul style="list-style-type: none"> ■ Insert on object: amContract ■ Insert on object: amCntrRent ■ PostUpdate on object: sePeriodicity ■ PostUpdate on object: mAmount ■ PostUpdate on object: mPayments ■ PostUpdate on object: sePeriodicity ■ PostUpdate on object: bMainRent 	<p>Ensures the consistency of the frequency of payment and the full amount between the contract and the main rent.</p>	

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
VersInitExpLine	<ul style="list-style-type: none"> ■ Insert on object: amContract ■ PostUpdate on object: mIntPay ■ PostUpdate on object: mIntPayAst ■ PostUpdate on object: dStart 	<p>If the initial payment of the contract, the total of initial payments for the assets on the contract or the contract start date is modified, this agent updates or creates the expense line corresponding to the initial payment of the contract. In particular, if there is a difference between the initial payment of the contract and the sum of initial payments of the assets on the contract a compensating expense line is created or updated.</p>	



14 Cost Centers table (amCostCenter)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Cost Centers table. Each section deals with a different type of automatic mechanism.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 14.1. Validity scripts on the table

Script	Description
<pre>If Not IsEmpty([dEnd]) and Not IsEmpty([dStart]) and [dStart] > [dEnd] Then Err.Raise(-2009, "The end date (dEnd) must be greater than or equal to the start date (dStart).") RetVal = FALSE Else RetVal = TRUE End If</pre>	<p>If the creation and end dates of the cost center are not empty and the end date comes before the creation date, the record is rejected.</p>

Table 14.2. Default value scripts

Object concerned	Script	Description
Code	<code>RetVal = "C" + AmCounter("amCostCenter_Code", 6)</code>	By default, the unique code of a cost center is the concatenation of the letter C and the value of the amCostCenter_Code counter on 6 figures.
dRecalcFrom	<code>RetVal = AmDate()</code>	By default, the date from which the expense lines of the cost center are to be split is the record creation date.
dRecalcTo	<code>RetVal = AmDate()</code>	By default, the date up until which the expense lines of the cost center are to be split is the record creation date.
dStart	<code>RetVal = AmDate()</code>	By default, the cost center creation date is the record creation date.

Integrity rules

There are no integrity rules on the Cost Centers table (**amCostCenter**).

Agents

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
FullName agent	<ul style="list-style-type: none"> ■ Insert in the am-CostCenter table ■ Post-Update on the Code field ■ Post-Update on the IParentId link ■ Pre-Update on the Code field ■ Post-Update on the IParentId link 	<p>This agent manages tree structures in hierarchic tables.</p> <p>In the Brands table, it maintains hierarchical integrity in the case of sub-cost centers. The full name of the brand and its hierarchical level are recalculated if:</p> <ul style="list-style-type: none"> ■ A cost center is created ■ The code of the cost center is modified ■ The parent cost center is modified 	<ul style="list-style-type: none"> ■ FullName ■ sLvl



15 | Departments and Employees table (amEmplDept)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Departments and Employees table. Each section deals with a different type of automatic mechanism.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 15.1. Default value scripts

Object concerned	Script	Description
BarCode	<code>RetVal = "U" + AmCounter("amEmplDept_BarCode", 6)</code>	By default, the barcode associated with a the employee or department is the concatenation of the letter U and the value of the amEmplDept_BarCode counter on 6 figures.
dHire	<code>RetVal = AmDate()</code>	By default, the hire date is the record creation date.
EMail	<code>RetVal = [Parent.EMail]</code>	By default, this field, which contains the e-mail of the employee or department, takes the same value as the e-mail of its parent.
Fax	<code>RetVal = [Parent.Fax]</code>	By default, this field, which contains the fax number of the employee or department, takes the same value as the fax number of its parent.
IDNo	<code>if [bDepartment]=0 Then RetVal = "U" + AmCounter("amEmplDept_BarCode", 6) End If</code>	In the case of employees only, by default, the employee ID is the concatenation of the letter U and the value of the amEmplDept_BarCode counter on 6 figures.
ICostId	<code>RetVal = [Parent.ICostId]</code>	By default, this field, which contains the identifier of the cost center of the employee or department, takes the same value as the identifier of its parent.
IIconId	<code>RetVal = [Parent.IIconId]</code>	By default, this field, which contains the identifier of the icon used to represent the department or employee, takes the same value as the identifier of its parent.

Object concerned	Script	Description
lLocald	<code>RetVal = [Parent.lLocaId]</code>	By default, this field, which contains the identifier of the location of the employee or department, takes the same value as the identifier of its parent.
lSupervld	<code>RetVal = [Parent.lSupervId]</code>	By default, this field, which contains the identifier of the supervisor of the employee or department, takes the same value as the identifier of its parent.
Phone	<code>RetVal = [Parent.Phone]</code>	By default, this field, which contains the telephone number of the employee or department, takes the same value as the telephone number of its parent.

Table 15.2. Irrelevance scripts

Object concerned	Script	Description
bAdminRight	<code>RetVal= (0<> [bDepartment])</code>	This field is irrelevant in the case of a department.
bCanReadArchive	<code>RetVal= (0<> [bDepartment])</code>	This field is irrelevant in the case of a department.
bHAdmin	<code>RetVal= (0<> [bDepartment])</code>	This field is irrelevant in the case of a department.
bIsRCHotliner	<code>RetVal= (0<> [bDepartment])</code>	This field is irrelevant in the case of a department.
bIsRCManager	<code>RetVal= (0<> [bDepartment])</code>	This field is irrelevant in the case of a department.
dHire	<code>RetVal= (0<> [bDepartment])</code>	This field is irrelevant in the case of a department.
dLeave	<code>RetVal= (0<> [bDepartment])</code>	This field is irrelevant in the case of a department.
FirstName	<code>RetVal= (0<> [bDepartment])</code>	This field is irrelevant in the case of a department.
FirstName2	<code>RetVal= (0<> [bDepartment])</code>	This field is irrelevant in the case of a department.
HomePhone	<code>RetVal= (0<> [bDepartment])</code>	This field is irrelevant in the case of a department.

Object concerned	Script	Description
Identifier	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
IDefCurld	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
ILoginActId	RetVal = (0<> [bDepartment]) OR (" = [UserLogin])	This field is only relevant in the case of a department or if the user login is empty.
LoginPassword	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
IPhotold	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
IProfileId	RetVal = (0<> [bDepartment] OR 0<> [bAdminRight])	This field is only relevant in the case of a department or if the user has administration rights.
ISupervId	RetVal = (0= [bDepartment])	This field is only relevant in the case of a department.
MailLogin	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
MailPassword	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
MobilePhone	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
MrMrs	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
seLoginClass	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
Title	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
UserDesc	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
UserDomain	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
UserLogin	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
UserName	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
Absences	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
DefCurrency	RetVal= (0<> [bDepartment])	This field is irrelevant in the case of a department.
EmplGroups	RetVal = (0<> [bDepartment])	This field is irrelevant in the case of a department.

Object concerned	Script	Description
Entitlement	<code>RetVal = (0<>[bDepartment])</code>	This field is irrelevant in the case of a department.
LoginAction	<code>RetVal = (0<>[bDepartment]) OR ("=[UserLogin])</code>	This field is only relevant in the case of a department or if the user login is empty.
Photo	<code>RetVal=(0<>[bDepartment])</code>	This field is irrelevant in the case of a department.
Profile	<code>RetVal = (0<>[bDepartment]) OR 0<>[bAdminRight])</code>	This field is only relevant in the case of a department or if the user has administration rights.
Supervisor	<code>RetVal = (0=[bDepartment])</code>	This field is only relevant in the case of a department.
Trainings	<code>RetVal=(0<>[bDepartment])</code>	This field is irrelevant in the case of a department.

Integrity rules

Name of the rule	List of monitored objects	Rule(s) verified	List of any modified objects
CPasswordInteg	<ul style="list-style-type: none"> ■ SyncRead on object: UserLogin ■ SyncRead on object: LoginPassword 	When an employee's login is updated, this integrity rule empties the password.	◆ LoginPassword

Name of the rule	List of monitored objects	Rule(s) verified	List of any modified objects
CPasswordInteg	<ul style="list-style-type: none"> ■ SyncRead on object: MailLogin ■ SyncRead on object: MailPassword 	When an employee's mail is updated, this integrity rule empties the password.	<ul style="list-style-type: none"> ◆ MailPassword

Agents

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CAdminLoginAgent	<ul style="list-style-type: none"> ■ PreDelete on object: amEmplDept ■ PreUpdate on object: UserLogin ■ PreUpdate on object: seLoginClass ■ PreUpdate on object: bAdminRight 	Forbids the Admin user from doing the following operations: <ul style="list-style-type: none"> ■ Archival ■ Deletion ■ Modifying login ■ Modifying login type ■ Modifying administration rights 	
CGBLoginNumber-Check	<ul style="list-style-type: none"> ■ PreUpdate on object: UserLogin ■ PreUpdate on object: seLoginClass 	Makes sure the number of named users is not exceeded. If this the case, the login being edited is turned into a concurrent user.	<ul style="list-style-type: none"> ◆ seLoginClass
CGbPerson2Service	<ul style="list-style-type: none"> ■ Insert on object: amEmplDept ■ PreUpdate on object: lParentId 	Changes an employee, created on the fly or imported, into a department if another employee or department is attached to them.	

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
FullName agent	<ul style="list-style-type: none"> ■ Insert on object: amEmplDept ■ PostUpdate on object: Name ■ PostUpdate on object: FirstName ■ PostUpdate on object: IDNo ■ PostUpdate on object: bDepartment ■ PostUpdate on object: IParentId ■ PreUpdate on object: Name ■ PreUpdate on object: FirstName ■ PreUpdate on object: IDNo ■ PreUpdate on object: bDepartment ■ PreUpdate on object: IParentId 	<p>This agent manages tree structures in hierarchic tables.</p> <p>In the Departments and Employees table, it ensures the consistency of the hierarchy. The full name of the employee or the service and their hierarchical level are recalculated if:</p> <ul style="list-style-type: none"> ■ the name of the employee or department is modified ■ the first name of the employee is modified ■ the Employee ID of the employee is modified ■ the employee record is converted to a department or vice-versa ■ its parent is modified 	<ul style="list-style-type: none"> ■ FullName ■ sLvl



16

Locations table (amLocation)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Locations table. Each section deals with a different type of automatic mechanism.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 16.1. Default value scripts

Object concerned	Script	Description
Address1	<code>RetVal = [Parent.Address1]</code>	By default, the address of a location is identical to that of its parent location.
Address2	<code>RetVal = [Parent.Address2]</code>	By default, the address of a location is identical to that of its parent location.
BarCode	<code>RetVal = "L" + AmCounter("amLocation_BarCode", 6)</code>	By default, the unique code of the location is the concatenation of the letter L and the value of the amLocation_BarCode counter on 6 figures.
City	<code>RetVal = [Parent.City]</code>	By default, the city of a location is identical to that of its parent location.
ICostId	<code>RetVal = [Parent.lCostId]</code>	By default, the cost center of a location is identical to that of its parent location.
ICountryId	<code>RetVal = [Parent.lCountryId]</code>	By default, the country of a location is identical to that of its parent location.
IconId	<code>RetVal = [Parent.lIconId]</code>	By default, this field, which contains the identifier of the icon used to represent the location, inherits the same value from its parent location.
IStockUsedId	<code>RetVal = [Parent.lStockUsedId]</code>	By default, the stock serving a location is identical to that of its parent location.
ITaxJurisId	<code>RetVal = [Parent.lTaxJurisId]</code>	By default, the jurisdiction of a location is identical to that of its parent location.
Name	<code>RetVal = "" if 0<>[lSocId] then RetVal = [Company.Name] end if</code>	By default, if the location is a company site, it inherits its name from the company. Otherwise, the name is left empty.
State	<code>RetVal = [Parent.State]</code>	By default, the state of a location is identical to that of its parent location.

Object concerned	Script	Description
ZIP	RetVal = [Parent.ZIP]	By default, the postal code of a location is identical to that of its parent location.

Integrity rules

There are no integrity rules on the Locations table (**amLocation**).

Agents

The following table lists the agents working on the Locations table (**amLocation**).

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
FullName agent	<ul style="list-style-type: none"> ■ Insert in the am-Location table ■ Post-Update on the Name field ■ Post-Update on the IParentId link ■ Pre-Update on the Name field ■ Pre-Update on IParentId link 	<p>This agent manages tree structures in hierarchic tables.</p> <p>In the Locations table, it maintains hierarchical integrity in the case of sub-locations. The full name of the location and its hierarchical level are recalculated if:</p> <ul style="list-style-type: none"> ■ A location is created ■ The name of the location is modified ■ The parent location is modified 	<ul style="list-style-type: none"> ■ FullName ■ sLvl

Workflows

The following table summarizes the workflows operating on the Locations table (**amLocation**).

Warning:

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Determining the workflows used for a table](#) [page 183] at the end of this document.

Workflow reference	Workflow type	Description
PROP_ADDR	Synchronous	This workflow is triggered if the address of a location is modified (Address1, Address2, City, Country, State, ZIP fields). It propagates the modifications to the sub-locations.



17 Models table (amModel)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Models table. Each section deals with a different type of automatic mechanism.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 17.1. Default value scripts

Object concerned	Script	Description
AcctCode	<code>RetVal = [Parent.AcctCode]</code>	By default, the accounting code of the model is that of the model.
BarCode	<code>RetVal = "M" + AmCounter("amModel_BarCode", 6)</code>	By default, the barcode of the model is the concatenation of the letter M and the value of the amModel_BarCode counter on 6 figures.
blInvent	<code>RetVal = 1</code>	By default, the model is inventories during barcode inventories.
Certification	<code>RetVal = [Parent.Certification]</code>	By default, the certification is inherited from the parent model.
fCountFactor	<code>RetVal = 1</code>	By default, the number of points to be counted by installation or utilization of the model is set to 1.
fRoundingQty	<code>RetVal = 0</code>	By default, no roundings are tolerated for the quantities linked to the model.
fUseQty	<code>RetVal = 1</code>	By default, the indivisible quantity of the model is set to 1. This quantity enables you to specify the fraction used to divide batches created from the model.
lBrandId	<code>RetVal = [Parent.lBrandId]</code>	By default, this field, which contains the identifier of the brand of the model, takes the same value as the identifier of the brand of the parent model.
lIconId	<code>RetVal = [Parent.lIconId]</code>	By default, this field, which contains the identifier of the icon used to represent the model, takes the same value as the identifier of the icon of the parent model.

Object concerned	Script	Description
lNatureId	<pre>RetVal = [Parent.lNatureId]</pre>	By default, this field, which contains the identifier of the nature of the model, takes the same value as the identifier of the nature of the parent model.
lUseUnitId	<pre>RetVal = [Parent.lUseUnitId]</pre>	By default, this field, which contains the identifier of the unit of the model, takes the same value as the identifier of the unit of the parent model.
Prefix	<pre>RetVal = [Parent.Prefix]</pre>	By default, the prefix of the model is that of the model.
pTaxRate	<pre>RetVal = 19.6/100 if [lParentId] <> 0 then RetVal = [Parent.pTaxRate] end if</pre>	By default, the applicable tax rate for the model is 7.75%. If the model has a parent model, it inherits its tax rate.
seContractType	<pre>RetVal = [Nature.seCtrType]</pre>	By default, the contract type associated with the model is inherited from the nature of the model.
seDevSdType	<pre>RetVal = 0</pre>	Used for Cable only. In this case, by default, the model represents a single-sided device.
seDevType	<pre>RetVal = 0</pre>	Used for Cable only. In this case, by default, the model represents an active device.
seSoftLicMulti	<pre>RetVal = 0</pre>	By default, software based on this model can be installed on one single computer.
seSoftLicType	<pre>RetVal=3</pre>	By default, the license type associated with the model is Not defined .

Table 17.2. Mandatory scripts

Object concerned	Script	Description
BarCode	<pre>RetVal = (0<>[bInvent])</pre>	This field must be populated if the model is to be inventoried in barcode inventories.

Table 17.3. Irrelevance scripts

Object concerned	Script	Description
bSpeaker	<code>RetVal = ("amPhone"<>[Nature.OverflowTbl])</code>	This field is only relevant if the nature of the model creates a telephone.
bVoiceMail	<code>RetVal = ("amPhone"<>[Nature.OverflowTbl])</code>	This field is only relevant if the nature of the model creates a telephone.
CableType	<code>RetVal = (8<>[Nature.seBasis])</code>	This field is only relevant if the model is intended to create a cable.
Certification	<code>RetVal = (0=[bRequestable])</code>	This field is only relevant if the model can be included in a purchase request.
ContractNature	<code>RetVal = (4<>[Nature.seBasis])</code>	This field is only relevant if the model is intended to create a contract.
CPUType	<code>RetVal = (1<>[Nature.seBasis]) OR ("amComputer"<>[Nature.OverflowTbl])</code>	This field is only relevant if the model is intended to create a portfolio item or if the nature of the model creates a computer.
dCertifEnd	<code>RetVal = (0=[bRequestable])</code>	This field is only relevant if the model can be included in a purchase request.
dCertification	<code>RetVal = (0=[bRequestable])</code>	This field is only relevant if the model can be included in a purchase request.
DeviceType	<code>RetVal = 1 ' Must be an Asset and a Device if [Nature.seBasis] = 1 and [Nature.bDevice] = 1 then RetVal = 0 end if</code>	This field is only relevant if the model is intended to create a portfolio item that is a cable device.
fCountFactor	<code>RetVal = ("amSoftInstall"<>[Nature.OverflowTbl])</code>	This field is only relevant if the nature of the model creates a software installation.

Object concerned	Script	Description
fRoundingQty	<pre>RetVal = 1 ' Must be a bulk asset or a Cable (length) if ([Nature.seBasis] = 1 a nd [Nature.seMgtConstraint]<>2) or ([Nature.seBasis] = 8) then RetVal = 0 end if</pre>	This field is only relevant if the model is intended to create a portfolio item whose management constraint is Free or Unique asset tag , or is a cable.
fUseQty	<pre>RetVal = 1 ' Must be a bulk asset or a Cable (length) if ([Nature.seBasis] = 1 a nd [Nature.seMgtConstraint]<>2) or ([Nature.seBasis] = 8) then RetVal = 0 end if</pre>	This field is only relevant if the model is intended to create a cable or a portfolio item that is a cable device.
InstLanguage	<pre>RetVal= ([Nature.seOverflow Tbl]<>3)</pre>	This field is only relevant if the nature of the model creates a software installation.
IColorCodeId	<pre>RetVal = (8<>[Nature.seBas is])</pre>	This field is only relevant if the model is intended to create a cable.
ICPUSpeedMHz	<pre>RetVal = (1<>[Nature.seBas is]) OR ("amComputer"<>[Na ture.OverflowTbl])</pre>	This field is only relevant if the model is intended to create a portfolio item or if the nature of the model creates a computer.
IDiskSizeMb	<pre>RetVal = (1<>[Nature.seBas is]) OR ("amComputer"<>[Na ture.OverflowTbl])</pre>	This field is only relevant if the model is intended to create a portfolio item or if the nature of the model creates a computer.
LicLanguage	<pre>RetVal= ([Nature.bSoftLicen se]=0)</pre>	This field is only relevant if the nature of the model creates a software license.
ILabelRuleId	<pre>RetVal = 1 ' Must be a cable or (an a sset and a device) if ([Nature.seBasis] = 8) or ([Nature.seBasis] = 1 a nd [Nature.bDevice] = 1) t hen RetVal = 0 end if</pre>	This field is only relevant if the model is intended to create a cable or a portfolio item that is a cable device.

Object concerned	Script	Description
IMemorySizeMb	<pre>RetVal = (1<>[Nature.seBasis]) OR ("amComputer"<>[Nature.OverflowTbl])</pre>	This field is only relevant if the model is intended to create a portfolio item or if the nature of the model creates a computer.
IPins	<pre>RetVal = 1 ' Must be an Asset and a Device if [Nature.seBasis] = 1 and [Nature.bDevice] = 1 then RetVal = 0 end if</pre>	This field is only relevant if the model is intended to create a portfolio item that is a cable device.
ISoftLicUseRights	<pre>RetVal = (0=[Nature.bSoftLicense])</pre>	This field is only relevant if the nature of the model creates a software license.
IUseUnitId	<pre>RetVal = 1 ' Must be a bulk asset or a Cable (length) if ([Nature.seBasis] = 1 and [Nature.seMgtConstraint]<>2) or ([Nature.seBasis] = 8) then RetVal = 0 end if</pre>	This field is only relevant if the model is intended to create a portfolio item whose management constraint is Free or Unique asset tag , or is a cable.
IWOCalendarId	<pre>RetVal = (3<>[Nature.seBasis])</pre>	This field is only relevant if the model is intended to create a work order.
seAuthorization	<pre>RetVal = ("amSoftInstall"<>[Nature.OverflowTbl])</pre>	This field is only relevant if the nature of the model creates a software installation.
seContractType	<pre>RetVal = (4<>[Nature.seBasis])</pre>	This field is only relevant if the model is intended to create a contract.
seDevSdType	<pre>RetVal = 1 ' Must be an Asset and a Device if [Nature.seBasis] = 1 and [Nature.bDevice] = 1 then RetVal = 0 end if</pre>	This field is only relevant if the model is intended to create a portfolio item that is a cable device.
seDevType	<pre>RetVal = 1 ' Must be an Asset and a Device if [Nature.seBasis] = 1 and [Nature.bDevice] = 1 then RetVal = 0 end if</pre>	This field is only relevant if the model is intended to create a portfolio item that is a cable device.

Object concerned	Script	Description
seSoftLicType	<code>RetVal = (0=[Nature.bSoftLicense])</code>	This field is only relevant if the nature of the model creates a software license.
seWOType	<code>RetVal = (3<>[Nature.seBasis])</code>	This field is only relevant if the model is intended to create a work order.
SoftMedia	<code>RetVal = (0=[Nature.bSoftLicense])</code>	This field is only relevant if the nature of the model creates a software license.
SoftOS	<code>RetVal = (0=[Nature.bSoftLicense])</code>	This field is only relevant if the nature of the model creates a software license.
tsCntrDuration	<code>RetVal = (4<>[Nature.seBasis])</code>	This field is only relevant if the model is intended to create a contract.
tsTrngDuration	<code>RetVal = (6<>[Nature.seBasis])</code>	This field is only relevant if the model is intended to create a training.
tsWOSchedFixDelay	<code>RetVal = (3<>[Nature.seBasis])</code>	This field is only relevant if the model is intended to create a work order.
tsWOSchedFixDur	<code>RetVal = (3<>[Nature.seBasis])</code>	This field is only relevant if the model is intended to create a work order.
VersionLevel	<code>RetVal = ("amSoftInstall"<>[Nature.OverflowTbl])</code>	This field is only relevant if the nature of the model creates a software installation.
WOPriority	<code>RetVal = (3<>[Nature.seBasis])</code>	This field is only relevant if the model is intended to create a work order.
ColorCode	<code>RetVal = (8<>[Nature.seBasis])</code>	This field is only relevant if the model is intended to create a cable.
FieldAdjustTempls	<code>RetVal = (99=[Nature.seBasis])</code>	This field is only relevant if the model creates nothing.
LabelRule	<code>RetVal = 1 ' Must be a cable or (an asset and a device) if ([Nature.seBasis] = 8) or ([Nature.seBasis] = 1 and [Nature.bDevice] = 1) then RetVal = 0 end if</code>	This field is only relevant if the model is intended to create a cable or a portfolio item that is a cable device.

Object concerned	Script	Description
LicenseSoftInfos	RetVal = (0=[Nature.bSoftLicense])	This field is only relevant if the nature of the model creates a software license.
ModelSlots	RetVal = 1 ' Must be an Asset and a Device if [Nature.seBasis] = 1 and [Nature.bDevice] = 1 then n RetVal = 0 end if	This field is only relevant if the model is intended to create a portfolio item that is a cable device.
Pairs	RetVal = (8<>[Nature.seBasis])	This field is only relevant if the model is intended to create a cable.
Ports	RetVal = 1 ' Must be connectable if [Nature.seBasis] = 1 and [Nature.bIsCnxClient] > 0 then RetVal = 0 end if	This field is only relevant if the model is intended to create a portfolio item that can be connected
SoftwareSoftInfos	RetVal = ("amSoftInstall"<>[Nature.OverflowTbl])	This field is only relevant if the nature of the model creates a software installation.
UseUnit	RetVal = 1 ' Must be a bulk asset or a Cable (length) if ([Nature.seBasis] = 1 and [Nature.seMgtConstraint]<>2) or ([Nature.seBasis] = 8) then RetVal = 0 end if	This field is only relevant if the model is intended to create a portfolio item whose management constraint is Free or Unique asset tag , or is a cable.
WOCalendar	RetVal = (3<>[Nature.seBasis])	This field is only relevant if the model is intended to create a work order.

Agents

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CBiSoftInteg	<ul style="list-style-type: none"> ■ ASyncRead on object: Nature.bSoftLicense ■ SyncRead on object: ISoftLicUseRights ■ SyncRead on object: seSoftLicType ■ SyncRead on object: seSoftLicMulti 	<p>The following rules are enforced for a model for which the nature is a software license:</p> <ul style="list-style-type: none"> ■ If the license is not Multiple-user (seSoftLicMulti), the number of users for the license (ISoftLicUserRights) is forced to 1. The license type (seSoftLicType) is forced to Per named workstation. ■ If the number of users of the license is greater than 1, the license becomes Multiple-user. 	<ul style="list-style-type: none"> ■ ISoftLicUseRights ■ seSoftLicMulti ■ seSoftLicType
COverflow-ChangeAgent	<ul style="list-style-type: none"> ◆ PreUpdate on object: INatureId 	<p>This agent stops the nature of a model from being changed if doing so implies the associated overflow table being changed also.</p>	

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
FullName agent	<ul style="list-style-type: none"> ■ Insert on object: amModel ■ PostUpdate on object: Name ■ PostUpdate on object: IParentId ■ PreUpdate on object: Name ■ PreUpdate on object: IParentId 	<p>This agent manages tree structures in hierarchic tables.</p> <p>In the Models table, it maintains the integrity of the hierarchical structure. The full name of the product and its hierarchical level are recalculated if:</p> <ul style="list-style-type: none"> ■ the name of the model is modified ■ its parent is modified 	<ul style="list-style-type: none"> ■ FullName ■ sLvl



18 | Portfolio Items table (amPortfolio)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Portfolio Items table. Each section deals with a different type of automatic mechanism.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 18.1. Validity scripts on the table

Script	Description
<pre>If IsEmpty([dAssignment]) and [seAssignment]=0 Then Err.Raise(-2009, "Since it is no longer in stock, you must specify an assignment (in-service date) for this asset.") RetVal = FALSE Else RetVal = TRUE End If</pre>	<p>If the item is In use (<code>[seAssignment]=0</code>), you must specify an in-service date. Otherwise the record is rejected.</p>

Table 18.2. Default value scripts

Object concerned	Script	Description
AssetTag	<code>RetVal = [Asset.AssetTag]</code>	By default, the asset tag of a portfolio item is that of the associated asset.
AvgPriceCur	<code>RetVal = AmDefaultCurrency()</code>	By default, this field is set to the value of the default currency.
bUseQty	<pre>if [Model.Nature.seMgtConstraint]=2 OR [lModelId]=0 Then RetVal = 0 else RetVal = 1 End if</pre>	If the management constraint of the nature of the model associated with the portfolio item is set to Unique asset tag or the identifier of the associated model is null, then the portfolio item does not have an associated quantity.
Code	<code>RetVal = AmCounter("amAssignment_Code", 6)</code>	By default, this field is set to the value of the amAssignment_Code counter on 6 figures.
dAssignment	<pre>' Do we assign it now ? if 1<>[seAssignment] then RetVal = AmDate() end if</pre>	If the portfolio item is not set to In stock , it is in service and the in-service date is populated with the current date.
dtAvgPriceCv	<code>RetVal = AmDate()</code>	By default, the conversion date of the unit value is the current date.

Object concerned	Script	Description
fQty	<pre>RetVal = 1 if 0<>[lAstId] then RetVal = (amDbGetDouble("S elect SUM(fTotalQty) FROM amAsset WHERE lAstId=" & [lAstId])) - (amDbGetDouble ("Select SUM(fQty) FROM am Portfolio WHERE lAstId=" & [lAstId])) end if</pre>	<ul style="list-style-type: none"> ■ When no asset is associated with the portfolio item, the number of units in the batch is set to 1. ■ When an asset is associated with the portfolio item, this field is set to the difference between the total quantity of units in the batch and the number of units in the batch.
lCostCatId	<pre>RetVal = [Model.lCostCatId]</pre>	<p>By default, the cost category associated with the portfolio item is that of the model.</p>
lCostId	<pre>if [lParentId]=0 Then RetVal = [User.lCostId] else RetVal = [Parent.lCostId] End if</pre>	<ul style="list-style-type: none"> ■ If the portfolio item has a parent item, its cost center is that of the parent. ■ Otherwise, the cost center is that of the user of the portfolio item.
lIconId	<pre>RetVal = [Model.lIconId]</pre>	<p>By default, this field, which contains the identifier of the icon used to represent the portfolio item, inherits the same value as that of the model from which it is derived.</p>

Object concerned	Script	Description
lLocald	<pre> if (0<>[lParentId]) AND (0 <>[Parent.lLocaId]) then RetVal = [Parent.lLocaId] elseif (0<>[lUserId]) AND (0<>[User.lLocaId]) then RetVal = [User.lLocaId] elseif (0<>[lStockId]) AND (0<>[Stock.lStockId]) then RetVal = [Stock.lLocaId] end if </pre>	<ul style="list-style-type: none"> ■ If the portfolio item has a parent item and a location is defined for the parent, then the location of the portfolio item is that of its parent. ■ If this is not the case and the portfolio item has a user with a defined location, then the location of item is set to that of its user. ■ Otherwise, if the item has a stock that is associated with a location, then the location of the item is that of the stock.
lModelId	RetVal = [Asset.lModelId]	By default, the model is that of the asset associated with the portfolio item.
lStockId	RetVal = [Location.lStockUsedId]	By default, the stock is that of the location of the portfolio item.
lSupervId	RetVal = [Parent.lSupervId]	By default, the supervisor is that of the parent portfolio item.
lUserId	RetVal = [Parent.lUserId]	By default, the user is that of the parent portfolio item.

Table 18.3. Mandatory scripts

Object concerned	Script	Description
lStockId	RetVal = (1 = [seAssignment])	If the Assignment of the portfolio item is In stock then is mandatory to specify a stock for the portfolio item.

Table 18.4. Read-Only scripts

Object concerned	Script	Description
fQty	<pre>RetVal = (2=[Model.Nature.seMgtConstraint] OR [ModelId]=0 OR [lAstId]<>0)</pre>	If the management constraint of the nature of the model associated with the portfolio item is set to Unique asset tag , then the number of units in the batch cannot be modified.

Table 18.5. Irrelevance scripts

Object concerned	Script	Description
AssetTag	<pre>RetVal = (2<>[Model.Nature.seMgtConstraint] OR [ModelId]=0)</pre>	This field is irrelevant if the item is not managed with a Unique asset tag . It is not displayed in this case.
bUseQty	<pre>RetVal = (2=[Model.Nature.seMgtConstraint] OR [ModelId]=0)</pre>	This field is irrelevant if the item is not managed with a Unique asset tag . It is not displayed in this case.
Folder	<pre>if [Model.Nature.OverflowTbl] = "amSoftInstall" then RetVal = 0 else RetVal = 1 end if</pre>	This field, which stores the name of the installation folder of the software, is irrelevant if the corresponding item is not a software installation.
lAstId	<pre>RetVal = (0=[lAstId] OR [fQty] <> [Asset.fTotalQty])</pre>	If there is not asset associated with the portfolio item or the number of units in the batch is different from the total number of units in a batch then the link to an asset is irrelevant.
lLocald	<pre>'Relevant when in stock or assigned RetVal = 0 if [seAssignment]<>0 and [seAssignment]<>1 then RetVal = 1 end if</pre>	The link to a reservation is only relevant if the portfolio item is In stock or In use .

Object concerned	Script	Description
IStockId	<pre>'Relevant when in stock or waiting to enter stock RetVal = 0 if [seAssignment]<>1 and [seAssignment]<>3 then RetVal = 1 end if</pre>	The link to a stock is only relevant if the portfolio item is In stock or Awaiting receipt.
IUserId	<pre>RetVal = (amEvalScript("Ir relevant", "Stock", "")=FA LSE OR [seAssignment]=2)</pre>	The link to a user is only relevant if the portfolio item is not In stock or Retired.
IWorkOrderId	<pre>RetVal = (1<>[Model.Nature .bConsumable])</pre>	The link to a work order is only relevant if the portfolio item is a consumable.
RMANumber	<pre>RetVal = [seAssignment]<>4</pre>	This field, which contains the RMA number, is only relevant if the Assignment field of the portfolio item is Return for maintenance.
AddOn	<pre>RetVal = (2<>[Model.Nature .seMgtConstraint] OR [lMod elId]=0)</pre>	This field is irrelevant if the item is not managed with a Unique asset tag. It is not displayed in this case.
Asset	<pre>RetVal = (0=[lAstId] OR [f Qty] <> [Asset.fTotalQty])</pre>	If there is not asset associated with the portfolio item or the number of units in the batch is different from the total number of units in a batch then this link is irrelevant.
Batch	<pre>RetVal = (0=[lAstId] OR [f Qty] <> [Asset.fTotalQty])</pre>	If there is not asset associated with the portfolio item or the number of units in the batch is different from the total number of units in a batch then the link is irrelevant.
Computer	<pre>RetVal = ("amComputer"<>[M odel.Nature.OverflowTbl])</pre>	This link is only relevant if the portfolio item is a computer.
Location	<pre>'Relevant when in stock or assigned RetVal = 0 if [seAssignment]<>0 and [seAssignment]<>1 then RetVal = 1 end if</pre>	The link to a reservation is only relevant if the portfolio item is In stock or In use.
Phone	<pre>RetVal = ("amPhone"<>[Mode l.Nature.OverflowTbl])</pre>	This link is only relevant if the portfolio item is a telephone.

Object concerned	Script	Description
Reservation	<pre>'Relevant when in stock or waiting to enter stock RetVal = 0 if [seAssignment]<>1 and [seAssignment]<>3 then RetVal = 1 end if</pre>	The link to a reservation is only relevant if the portfolio item is In stock or Awaiting receipt.
Slot	<pre>RetVal = 1 ' Must be an asset and a d evice if [Model.Nature.seBasis] = 1 and [Model.Nature.bDe vice] > 0 then RetVal = 0 end if</pre>	This link to the available slots of a portfolio item is only relevant if the portfolio item is a cable device.
SoftInstall	<pre>RetVal = ("amSoftInstall"< >[Model.Nature.OverflowTbl])</pre>	This link is only relevant if the portfolio item is a software installation.
Stock	<pre>'Relevant when in stock or waiting to enter stock RetVal = 0 if [seAssignment]<>1 and [seAssignment]<>3 then RetVal = 1 end if</pre>	The link to a stock is only relevant if the portfolio item is In stock or Awaiting receipt.
User	<pre>RetVal = (amEvalScript("Ir relevant", "Stock", "")=FA LSE OR [seAssignment]=2)</pre>	The link to a user is only relevant if the portfolio item is not In stock or Retired.
WorkOrder	<pre>RetVal = (1<>[Model.Nature .bConsumable])</pre>	The link to a work order is only relevant if the portfolio item is a consumable.

Integrity rules

There are no integrity rules on the Portfolio Items table (**amPortfolio**).

Agents

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CAssignment-MergeAgent			None in records in the amPortfolio table. Depending on the operations performed by the agent, record may however be created or deleted.

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
		<p>This agent makes sure there are no two identical portfolio items in the database. The comparison is performed on all fields except the following:</p> <ul style="list-style-type: none"> ■ fQty ■ lPortfolioItemId ■ Code ■ FullName ■ dtLastModif ■ mAvgPrice ■ bCreatedOn-TheFly <p>On the basis of this comparison, if two identical portfolio items are found, the agents merges them into one single portfolio item and updates the quantity (fQty) and unit price (mAvgPrice).</p> <p>Note:</p> <p>The comparison also takes into account the features linked to the portfolio items. Two portfolio items that only differ in terms of a feature value are not considered to be the same.</p>	

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
	<ul style="list-style-type: none"> ■ Insert on object: amPortfolio ■ PostUpdate on object: AssetTag ■ PostUpdate on object: bUseQty ■ PostUpdate on object: dAssignment ■ PostUpdate on object: dtInvent ■ PostUpdate on object: Folder ■ PostUpdate on object: AvgPriceCur ■ PostUpdate on object: RMANumber ■ PostUpdate on object: seAssignment ■ PostUpdate on object: sLvl ■ PostUpdate on object: lAstId ■ PostUpdate on object: lParentId ■ PostUpdate on object: lCommentId ■ PostUpdate on object: lCostCatId ■ PostUpdate on object: lCostId ■ PostUpdate on object: lIconId ■ PostUpdate on object: lLocald ■ PostUpdate on object: lModelId ■ PostUpdate on object: lStockId ■ PostUpdate on ob- 		

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
	ject: ISupervId <ul style="list-style-type: none"> ■ PostUpdate on object: IUserId ■ PostUpdate on object: IWorkOrderId 		
CAssignmentParentAgent	<ul style="list-style-type: none"> ■ Insert on object: amPortfolio ■ PostUpdate on object: IParentId 	This agent performs the following operations: <ul style="list-style-type: none"> ■ It makes sure that the parent of a portfolio item is always an asset. An error is returned if this is not the case. ■ If the portfolio item has software installations as child records, it makes sure that the nature of the model of the portfolio item is flagged Has software installed. An error is returned if this is not the case. ■ If a portfolio item is a consumable, it makes sure that if it is linked to an asset, the assignment is set to Retired (or consumed). The assignment date is set to the current date. A consumption line is created and linked to both the portfolio item and its corresponding asset. 	<ul style="list-style-type: none"> ■ seAssignment ■ dAssignment

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CBatchQtyAgent	<ul style="list-style-type: none">■ Insert on object: amPortfolio■ PostUpdate on object: fTotalQty■ PostUpdate on object: fQty■ PostUpdate on object: lAstId■ PreDelete on object: amPortfolio■ PreDelete on object: amAsset	This agent maintains the consistency between the total quantity of a batch (fTotalQty) and the sum of the quantities of the batch items (fQty).	

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CGbAcquiDepAgent	<ul style="list-style-type: none"> ■ Insert on object: amAsset ■ Insert on object: amPortfolio ■ PostUpdate on object: dAcquisition ■ PostUpdate on object: mPrice ■ PostUpdate on object: mTax ■ PostUpdate on object: dIntPay ■ PostUpdate on object: mIntPay ■ PostUpdate on object: mIntPayTax ■ PostUpdate on object: seAcquMethod 	<p>This agent updates the expense lines associated with the portfolio item.</p> <p>It functions when a portfolio item is created or the following data items are updated for an existing portfolio item:</p> <ul style="list-style-type: none"> ■ seAcquMethod ■ dAcquisition ■ mPrice ■ mTax ■ mIntPay ■ mIntPayTax ■ dIntPay 	None in the amPortfolio table.
CGbAssetAssignment	<ul style="list-style-type: none"> ■ Insert on object: amPortfolio ■ PostDelete on object: amPortfolio ■ PreUpdate on object: lModelId 	<p>In case of creation of a portfolio item, if necessary this agent creates the corresponding record in the Assets table and the appropriate record in the overflow table matching the management constraint associated with the model of the portfolio item.</p>	None in the amPortfolio table.

Note:

The agent takes into account the distribution (split-billing) of expenses to the cost centers and cost categories. It may therefore create multiple expense lines.

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CGBsousBienIntegriteAgent	<ul style="list-style-type: none"> ■ PostUpdate on object: ILocald ■ PostUpdate on object: IUserId ■ PostUpdate on object: ISupervId ■ PostUpdate on object: IStockId ■ PostUpdate on object: seAssignment 	If one of the objects monitored for a portfolio item is updated, the agent propagates the modifications to all the child records.	None in the amPortfolio table.
CGBsousBienIntegriteAgent2	<ul style="list-style-type: none"> ◆ PreUpdate on object: IParentId 	<p>If the parent record of a portfolio item is modified, then the agent propagates the values of the following fields from the new parent record to the portfolio item:</p> <ul style="list-style-type: none"> ■ ILocald ■ IStockId ■ ISupervId ■ IUserId ■ seAssignment 	<ul style="list-style-type: none"> ■ ILocald ■ IStockId ■ ISupervId ■ IUserId ■ seAssignment

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CGbStockInOutAgent	<ul style="list-style-type: none">■ PreUpdate on object: seAssignment■ PreUpdate on object: IStockId■ PreUpdate on object: IUserId		<ul style="list-style-type: none">■ fQty■ ILocald■ IStockId■ ISupervId■ IUserId■ seAssignment

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
		<p>The agent performs the following operations:</p> <ul style="list-style-type: none"> ■ If the assignment of the portfolio item changes from In stock, the link pointing to the stock is emptied. ■ If the item has a stock and its assignment is In use, the agent changes it to In stock. ■ If the item has a stock and its assignment is set to Return for maintenance, Return to supplier or Missing, the agent returns an error. ■ If the item is In stock or Awaiting receipt and has no assigned stock, the agent changes the assignment to In use. ■ If the item is In stock or Awaiting receipt and has no assigned location, the agent sets it to that of the stock. ■ If the portfolio item does not have a location or is not In use and is assigned to a user, the agent sets its assignment to In 	

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
		<p>use and gives it the location of the user.</p> <ul style="list-style-type: none"> ■ If the assignment of the asset is not In use, the agent empties the User and Supervisor fields. ■ If the assignment of the asset changes from In stock to In use, the agent propagates the User from the reservation. ■ If the assignment of the portfolio item is set to In stock, the agent deletes any remaining reservations associated with the portfolio item. 	
COverflow-ChangeAgent	<ul style="list-style-type: none"> ◆ PreUpdate on object: IModelId 	<p>This agent stops the model of a portfolio item from being changed if doing so implies the associated overflow table being changed also.</p>	

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CRedundancyAgent	<ul style="list-style-type: none"> ■ Insert on object: amComputer ■ Insert on object: amPortfolio ■ PostUpdate on object: ItemId ■ PostUpdate on object: AssetTag ■ PostUpdate on object: AssetTag 	<p>This agent makes sure that the AssetTag fields of a computer and its associated portfolio item are identical:</p> <ul style="list-style-type: none"> ■ If the AssetTag field of a record in the amComputer table is modified, the agent propagates this change to the AssetTag field of the record in the corresponding amPortfolio table. ■ If the AssetTag field of a record in the amPortfolio table is modified, the agent propagates this change to the AssetTag field of the record in the corresponding amComputer table. 	<ul style="list-style-type: none"> ◆ AssetTag

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CRedundancyAgent	<ul style="list-style-type: none"> ■ Insert on object: amAsset ■ Insert on object: amPortfolio ■ PostUpdate on object: IModelId ■ PostUpdate on object: IModelId ■ PostUpdate on object: IAstId 	<p>This agent makes sure that an asset and its associated portfolio item always point to the same model:</p> <ul style="list-style-type: none"> ■ If the IModelId link of a record in the amAsset table is modified, the agent propagates this change to the IModelId field of the record in the corresponding amPortfolio table. ■ If the IModelId link of a record in the amPortfolio table is modified, the agent propagates this change to the IModelId link of the record in the corresponding amAsset table. 	<ul style="list-style-type: none"> ◆ IModelId

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CRedundancyAgent	<ul style="list-style-type: none"> ■ Insert on object: amAsset ■ Insert on object: amPortfolio ■ PostUpdate on object: AssetTag ■ PostUpdate on object: AssetTag ■ PostUpdate on object: IAstId 	<p>This agent makes sure that the AssetTag fields of an asset and its associated portfolio item are identical:</p> <ul style="list-style-type: none"> ■ If the AssetTag field of a record in the amAsset table is modified, the agent propagates this change to the AssetTag field of the record in the corresponding amPortfolio table. ■ If the AssetTag field of a record in the amPortfolio table is modified, the agent propagates this change to the AssetTag field of the record in the corresponding amAsset table. 	<ul style="list-style-type: none"> ◆ AssetTag
CReturnAssignmentAgent	<ul style="list-style-type: none"> ◆ PostUpdate on object: seAssignment 	<p>When the assignment of a portfolio item (which is not a consumable) is set to Return to supplier or Retired (or consumed), this item is de-hierarchized.</p>	

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
FullName agent	<ul style="list-style-type: none"> ■ Insert on object: amPortfolio ■ PostUpdate on object: Code ■ PostUpdate on object: IParentId ■ PreUpdate on object: Code ■ PreUpdate on object: IParentId 	<p>This agent manages tree structures in hierarchic tables.</p> <p>In the Portfolio items table, it maintains hierarchical integrity.</p> <p>The full name of the portfolio item and its hierarchical level are recalculated if:</p> <ul style="list-style-type: none"> ■ the portfolio item's code is changed ■ its parent is modified 	<ul style="list-style-type: none"> ■ FullName ■ sLvl

Workflows

The following tables summarize the workflows dealing with the Assets table (**amAsset**).

Warning:

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Determining the workflows used for a table](#) [page 183] at the end of this document.

Workflow reference	Workflow type	Description
BST_SAM04	Synchronous	When a portfolio item is retired (its assignment changes to Retired), this workflow asks the administrator if the licenses that were linked to this item can be freed up or not. A wizard is available to help in selecting the licenses to be freed.



19 Projects table (amProject)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Projects table. Each section deals with a different type of automatic mechanism.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 19.1. Validity scripts on the table

Script	Description
<pre>If Not IsEmpty([dEnd]) and Not IsEmpty([dStart]) and [dStart] > [dEnd] Then Err.Raise(-2009, "The end date (dEnd) must be greater than or equal to the start date (dStart).") RetVal = FALSE Else RetVal = TRUE End If</pre>	<p>If the start and end dates of the project are not empty and the end date comes before the start date, the record is rejected.</p>

Table 19.2. Default value scripts

Object concerned	Script	Description
Code	<code>RetVal = "C" + AmCounter("amProject_Code", 6)</code>	By default, the unique code of a project is the concatenation of the letter C and the value of the amProject_Code counter on 6 figures.
dStart	<code>RetVal = AmDate()</code>	By default, the start date of the project is the date of creation of the record.

Agents

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
CDateAlarmAgent	<ul style="list-style-type: none"> ◆ Post-Update on the dEnd object 	This agent recalculates the alarms associated with the project end date.	None in the amProject table.



20 | Stocks table (amStock)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Stocks table. Each section deals with a different type of automatic mechanism.

 **Note:**

There are no automatic mechanisms other than the default script values on this table.

Scripts

The following tables summarize the objects to which the scripts are attached and describe the operations performed by the scripts.

 **Warning:**

This section lists all the standard scripts touching upon the objects in the table concerned. This list cannot include any customizations and modifications specific to your own implementation of AssetCenter. To learn how to extract the scripts really used in your implementation concerning this table, refer to the appendix [Extracting all the scripts from a database](#) [page 173] at the end of this document.

Table 20.1. Default value scripts

Object concerned	Script	Description
Code	<code>RetVal = "C" + AmCounter("amStock_Code", 6)</code>	By default, the unique code of the stock is the concatenation of the letter C and the value of the amStock_Code counter on 6 figures.
DeliveryAddr	<code>RetVal = [Location.Address1] + " " + [Location.Address2] + " " + [Location.ZIP] + " " + [Location.City] + " " + [Location.State] + " " + [Location.Country.Name]</code>	By default, the delivery address for the stock is the concatenation of the address, postal code, city, state and country of the location associated with the stock.
dtValueCv	<code>RetVal = AmDate()</code>	By default, the conversion date for the stock value is the stock creation date.
ValueCur	<code>RetVal = AmDefaultCurrency()</code>	By default, the currency used to express the value of the stock is the default currency.



21 | Third-Party Companies table (amThirdParty)

CHAPTER

This chapter provides an exhaustive list of all the mechanisms dealing with the Third-Party Companies table. Each section deals with a different type of automatic mechanism.

 Note:

There are no automatic mechanisms other than the agents on this table.

Integrity rules

There are no integrity rules on the Third-Party Companies table (**amThirdParty**).

Agents

SQL name of the agent	List of monitored objects	Operations performed	List of any modified objects
IContactId_ICpyId	<ul style="list-style-type: none">▪ SyncRead on object: IContactId▪ SyncRead on object: ICpyId		<ul style="list-style-type: none">▪ IContactId▪ ICpyId



22 | Glossary

CHAPTER

Database terms

Stored procedure

Stored procedures enable you to move the burden of certain processes to the database engine rather than issue SQL statements from the client application. In practice, a stored procedure is unit of processing that receives parameters, executes operations and returns a result. They are written in a procedural language that includes SQL and are saved at the database server level.

Transaction

A transaction may be defined as a series of operations that are performed in full, or not at all, but never in part. If one of the operations fails then all the operations are cancelled. For example, if you wish to transfer a person from the Departments and Employees table to the Contracts table, the person must be first inserted into the Contracts table then deleted from the Employees table. It cannot be allowed for the second operation to be neglected otherwise the database would become inconsistent. From a practical point of view, a transaction is initiated by

a SQL statement and any modifications made to the database or only visible inside the transaction. They only become effective once the transaction has been validated by SQL operation called a **Commit**. If an anomaly occurs, all modifications can be cancelled by finishing the transaction with a **Rollback** command.

A transaction has the four following properties, which are universally recognized the domain of database engines:

- 1 **Atomicity:** A transaction is a unit of processing that so called atomic. Either it is performed in full, or not at all.
- 2 **Integrity:** A transaction changes the database from one consistent state to another. During the time of the transaction, the database remains unchanged.
- 3 **Isolation:** Modifications made as part of a transaction are invisible (in particular, to other transactions) for so long as they are not committed.
- 4 **Permanence:** After the **Commit** is reached in a transaction, the modifications are permanent and cannot be cancelled.

Trigger

A trigger associates a process with a specific action on the database. When the action is performed and the data matches a certain condition, the process is executed automatically by the database server. A systematic process is generally linked with an integrity constraint.

A trigger is specific type of stored procedure.

Exclusive lock

An exclusive lock is held by a transaction in order to exclude any other manipulations of the object or data locked.



A | Extracting all the scripts from a database

APPENDIX

This appendix aims to help you extract all the scripts included in your AssetCenter implementation.

AssetCenter Database Administrator, shipped with AssetCenter provides a template-based method to extract information (. **tpl** extension files).

Among the standard templates provided with AssetCenter, one of them, the **dbdict.tpl** file, enables you to export all the customization information from your database (including information on features, calculated features, configuration scripts, etc.) to a standard text-formatted file. Used along with a source control tool, this description file can be very useful for keeping a trace of all customization modifications made to the database.

This appendix includes a simplified template that just extracts information related to the script. You can copy the contents to a file with the . **tpl** extension and execute it in AssetCenter Database Administrator.

 Note:

For further information on templates, refer to the **Administration** guide, chapter **Standard database description files**.

Executing a template in AssetCenter Database Administrator

To execute a template in AssetCenter Database Administrator, use the following procedure:

- 1 Start AssetCenter Database Administrator if it is not already running and connect to your database,
- 2 Select **Action/ Templates/ Select folder** and select the folder containing the template or templates you wish to execute,
- 3 Select **Action/ Templates/ Refresh list**. The list of available templates is displayed in the second section of the **Action/ Templates** menu.
- 4 Execute the script of your choice by selecting **Action/ Templates**, and then the name of the script.

Examples of templates

The two following templates extract the information related to scripts. The first template saves the information in the form of an XML file (one XML file per table) using the DocBook format, the second in classic HTML format (one HTML file per table).

XML version

```
$ Desc: Scripts catalog XML - English - AssetCenter/InfraCenter
$ Type: XML
$ (c) Peregrine Systems, Inc. Documentation Team - 2004
$ Maintainer: Stéphane Bline
$ Warning: Do not modify this file directly. Send a formal change request
to me.
$OutputDir = $(Output.Path)
$MkDir($(OutputDir) + "tables")
$for Tables sort (SqlName ASC)
$SetOutput($(OutputDir) + "\tables\" + $(SqlName) + ".xml")
$TableSQLName=$(SqlName)
$ Output for the tables
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE sect1 PUBLIC "-//Norman Walsh//DTD DocBk XML V3.1.7//EN" "docboo
```

```

kx.dtd">
<sect1 lang="en" id="$(SqlName)"><title id="$(SqlName).Title">Scripts on t
able $(SqlName) ($(Label))</title>

$if ($(IsValidScript.CalcMode) = 2)
<sect2 id="SB-190919"><title id="SB-190920">Validity script on table $(Sql
Name)</title>
<programlisting id="SB-190921">$ReplaceChars($ReplaceChars($ScriptFormat($
(IsValidScript.Source),4),"&", "&"), "<", "<")</programlisting id="SB-1909
22">
</sect2>
$endif

<sect2 id="SB-190923"><title id="SB-190924">Scripts on fields</title>
$TableSQLName=$(SqlName)
$TableLabel=$(Label)
$for Fields sort (SqlName ASC)
$if ($(ReadOnlyScript.CalcMode) = 2) or ($(HistoryScript.CalcMode) = 2) o
r
($ (MandatoryScript.CalcMode) = 2) or ($(DefaultScript.Source)< id="SB-1909
25">"") or ($(RelevantScript.CalcMode)
=
2)
<sect3 lang="en" id="$(TableSQLName).$(SqlName)"><title id="$(TableSQLName
).$(SqlName).Title">Field $(SqlName) ($(Label))</title>
<informaltable id="SB-190926">
<tgroup cols="2" id="SB-190927">
<colspec colnum="1" colname="col1" colwidth="1*"/>
<colspec colnum="2" colname="col2" colwidth="1*"/>
<thead id="SB-190928">
<row id="SB-190929">
<entry colname="col1" align="center" id="SB-190930"><emphasis>Property</em
phasis></entry>
<entry colname="col2" align="center" id="SB-190931"><emphasis>Value</empha
sis></entry>
</row>
</thead>
<tbody id="SB-190932">
<row id="SB-190933">
<entry colname="col1" id="SB-190934"><emphasis>SQL name</emphasis></entry>
<entry colname="col2" align="center" id="SB-190935">$(SqlName)</entry>
</row>
<row id="SB-190936"><entry colname="col1" id="SB-190937"><emphasis>Name</e
mpphasis></entry>
<entry colname="col2" align="center" id="SB-190938">$(Label)</entry>
</row>
$if ($(ReadOnlyScript.Source)< id="SB-190939">"")
<row id="SB-190940"><entry colname="col1" id="SB-190941"><emphasis>Read-on
ly script</emphasis></entry>
<entry colname="col2" align="left" id="SB-190942"><programlisting id="SB-1
90943">$ReplaceChars($ReplaceChars($(ReadOnlyScript.Source),"&", "&"), "<
", "<")</programlisting id="SB-190944"></entry>
</row>
$endif
$if ($(HistoryScript.Source)< id="SB-190945">"")
<row id="SB-190946"><entry colname="col1" id="SB-190947"><emphasis>History

```

```

script</emphasis></entry>
<entry colname="col2" align="left" id="SB-190948"><programlisting id="SB-1
90949">$ReplaceChars($ReplaceChars($(HistoryScript.Source),"&", "&"), "<",
"<")</programlisting id="SB-190950"></entry>
</row>
<endif
$endif
$if ($(MandatoryScript.Source)< id="SB-190951">")
<row id="SB-190952"><entry colname="col1" id="SB-190953"><emphasis>Mandato
ry script</emphasis></entry>
<entry colname="col2" align="left" id="SB-190954"><programlisting id="SB-1
90955">$ReplaceChars($ReplaceChars($(MandatoryScript.Source),"&", "&"), "<
", "<")</programlisting id="SB-190956"></entry>
</row>
<endif
$endif
$if ($(DefaultScript.Source)< id="SB-190957">")
<row id="SB-190958"><entry colname="col1" id="SB-190959"><emphasis>Default
value script</emphasis></entry>
<entry colname="col2" align="left" id="SB-190960"><programlisting id="SB-1
90961">$ReplaceChars($ReplaceChars($(DefaultScript.Source),"&", "&"), "<",
"<")</programlisting id="SB-190962"></entry>
</row>
<endif
$endif
$if ($(RelevantScript.Source)< id="SB-190963">")
<row id="SB-190964"><entry colname="col1" id="SB-190965"><emphasis>Relevan
ce script</emphasis></entry>
<entry colname="col2" align="left" id="SB-190966"><programlisting id="SB-1
90967">$ReplaceChars($ReplaceChars($(RelevantScript.Source),"&", "&"), "<
", "<")</programlisting id="SB-190968"></entry>
</row>
<endif
</tbody>
</tgroup>
</informaltable>
</sect3>
<endif
<endfor
</sect2>

<sect2 id="SB-190969"><title id="SB-190970">Scripts on links</title>
$TableSQLName=$(SqlName)
$TableLabel=$(Label)
$for Links sort (SqlName ASC)
$if ($(RelevantScript.CalcMode) = 2)
<sect3 lang="en" id="$(TableSQLName).$(SqlName)"><title id="$(TableSQLName)
. $(SqlName).Title">Link $(SqlName) $(Label)</title>
<informaltable id="SB-190971">
<tgroup cols="2" id="SB-190972">
<colspec colnum="1" colname="col1" colwidth="1*"/>
<colspec colnum="2" colname="col2" colwidth="1*"/>
<thead id="SB-190973">
<row id="SB-190974">
<entry colname="col1" align="center" id="SB-190975"><emphasis>Property</em
phasis></entry>
<entry colname="col2" align="center" id="SB-190976"><emphasis>Value</empha
sis></entry>
</row>

```



```

</thead>
<tbody id="SB-190977">
<row id="SB-190978">
<entry colname="col1" id="SB-190979"><emphasis>SQL name</emphasis></entry>
<entry colname="col2" align="center" id="SB-190980">$(SqlName)</entry>
</row>
<row id="SB-190981"><entry colname="col1" id="SB-190982"><emphasis>Name</e
mpphasis></entry>
<entry colname="col2" align="center" id="SB-190983">$(Label)</entry>
</row>
$if ($(RelevantScript.Source)< id="SB-190984">")
<row id="SB-190985"><entry colname="col1" id="SB-190986"><emphasis>Relevan
ce script</emphasis></entry>
<entry colname="col2" align="left" id="SB-190987"><programlisting id="SB-1
90988">$ReplaceChars($ReplaceChars($(RelevantScript.Source),"&", "&"), "<
", "<")</programlisting id="SB-190989"></entry>
</row>
$endif
</tbody>
</tgroup>
</informaltable>
</sect3>
$endif
$endif
</sect2>
</sect1>
$endif

$script
'-----
' Format a script to put it in a cfg
'-----
Function ScriptFormat(strMemos as String, iSpace as Integer) as String
ScriptFormat = ReplaceChars(strMemos, Chr(10), Chr(10) + Space(iSpace))
End Function

'-----
' Replaces a string with another one
'-----
Function ReplaceChars(strMemos as String, strToRep as String, strReplaceme
nt as String) as String
Dim I as Integer
ReplaceChars = strMemos
I = InStr(0, ReplaceChars, strToRep)
While (I < id="SB-190990"> 0)
ReplaceChars = Left(ReplaceChars, I - 1) + strReplacement + Mid(ReplaceCha
rs, I +
Len(strToRep), Len(ReplaceChars))
I = InStr(I + Len(strToRep), ReplaceChars, strToRep)
Wend
End Function
$endscript

```

HTML version

```

$ Desc: Scripts catalog HTML - English - AssetCenter/InfraCenter
$ Type: HTML
$ (c) Peregrine Systems, Inc. Documentation Team - 2004
$ Maintainer: Stéphane Bline
$ Warning: Do not modify this file directly. Send a formal change request
to me.
$OutputDir = $(Output.Path)
$Mkdir($(OutputDir) + "tables")
$for Tables sort (SqlName ASC)
$SetOutput($(OutputDir) + "\tables\" + $(SqlName) + ".htm")
$TableSQLName=$(SqlName)
$ Output for the tables
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head id="SB-190994">
<title id="SB-190995">Scripts on table $(SqlName) ($(Label))</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" id
="SB-190996">
</head>
<body id="SB-190997">

$if ($(IsValidScript.CalcMode) = 2)
<h1 style="FONT-WEIGHT: bold; FONT-SIZE: 18pt; COLOR: #000066; FONT-FAMILY
: Verdana" id="SB-190998">Validity script on table $(SqlName)</h1>
<p style="font-family : Courier New; text-align : left; border : thin groo
ve" id="SB-190999">${ReplaceChars($ReplaceChars($ScriptFormat($(IsValidScri
pt.Source),4),"&", "&"), "<", "<")</p id="SB-191000">
$endif

<h1 style="FONT-WEIGHT: bold; FONT-SIZE: 18pt; COLOR: #000066; FONT-FAMILY
: Verdana" id="SB-191001">Scripts on fields</h1>
$TableSQLName=$(SqlName)
$TableLabel=$(Label)
$for Fields sort (SqlName ASC)
$if ($(ReadOnlyScript.CalcMode) = 2) or ($(HistoryScript.CalcMode) = 2) o
r ($(MandatoryScript.CalcMode) = 2) or ($(DefaultScript.Source)< id="SB-19
1002">"") or ($(RelevantScript.CalcMode) = 2)
<h2 style="FONT-WEIGHT: bold; FONT-SIZE: 10pt; COLOR: #000066; FONT-FAMILY
: Verdana; align: left" id="SB-191003">Field $(SqlName) ($(Label))</h2>
<table style="BORDER-RIGHT: #000066 1px solid; BORDER-TOP: #000066 1px sol
id; MARGIN-BOTTOM: 10px; BORDER-LEFT: #000066 1px solid; WIDTH: 400px; BOR
DER-BOTTOM: #000066 1px solid; table-width: 400px" id="SB-191004">
<tr style="PADDING-RIGHT: 2px; PADDING-LEFT: 2px; FONT-WEIGHT: bold; FONT-
SIZE: 8pt; PADDING-BOTTOM: 2px; COLOR: #ffffff; PADDING-TOP: 2px; FONT-FAM
ILY: Verdana, Helvetica, sans-serif; BACKGROUND-COLOR: #000066" id="SB-191
005">
<th id="SB-191006">Property</th>
<th id="SB-191007">Value</emphais></th>
</tr>
<tr style="PADDING-RIGHT: 2px; PADDING-LEFT: 2px; FONT-SIZE: 8pt; PADDING-
BOTTOM: 2px; VERTICAL-ALIGN: top; COLOR: #000066; PADDING-TOP: 2px; FONT-F
AMILY: Verdana, Tahoma, Helvetica, sans-serif; BACKGROUND-COLOR: #ffffff"

```

```

id="SB-191008">
<td id="SB-191009">SQL name</td>
<td id="SB-191010">$(SqlName)</td>
</tr>
<tr style="PADDING-RIGHT: 2px; PADDING-LEFT: 2px; FONT-SIZE: 8pt; PADDING-
BOTTOM: 2px; VERTICAL-ALIGN: top; COLOR: #000066; PADDING-TOP: 2px; FONT-F
AMILY: Verdana, Tahoma, Helvetica, sans-serif; BACKGROUND-COLOR: #ffffff"
id="SB-191011">
<td id="SB-191012">Name</td>
<td id="SB-191013">$(Label)</td>
</tr>
$if ($(ReadOnlyScript.Source)< id="SB-191014">""
<tr style="PADDING-RIGHT: 2px; PADDING-LEFT: 2px; FONT-SIZE: 8pt; PADDING-
BOTTOM: 2px; VERTICAL-ALIGN: top; COLOR: #000066; PADDING-TOP: 2px; FONT-F
AMILY: Verdana, Tahoma, Helvetica, sans-serif; BACKGROUND-COLOR: #ffffff"
id="SB-191015">
<td id="SB-191016">Read-only script</td>
<td id="SB-191017"><p style="font-family : Courier New; text-align : left;
border : thin groove" id="SB-191018">$ReplaceChars($ReplaceChars($(ReadOnl
yScript.Source),"&", "&"), "<", "<")</p id="SB-191019"></td>
</tr>
$endif
$if ($(HistoryScript.Source)< id="SB-191020">""
<tr style="PADDING-RIGHT: 2px; PADDING-LEFT: 2px; FONT-SIZE: 8pt; PADDING-
BOTTOM: 2px; VERTICAL-ALIGN: top; COLOR: #000066; PADDING-TOP: 2px; FONT-F
AMILY: Verdana, Tahoma, Helvetica, sans-serif; BACKGROUND-COLOR: #ffffff"
id="SB-191021">
<td id="SB-191022">History script</td>
<td id="SB-191023"><p style="font-family : Courier New; text-align : left;
border : thin groove" id="SB-191024">$ReplaceChars($ReplaceChars($(History
Script.Source),"&", "&"), "<", "<")</p id="SB-191025"></td>
</tr>
$endif
$if $(MandatoryScript.Source)< id="SB-191026">""
<tr style="PADDING-RIGHT: 2px; PADDING-LEFT: 2px; FONT-SIZE: 8pt; PADDING-
BOTTOM: 2px; VERTICAL-ALIGN: top; COLOR: #000066; PADDING-TOP: 2px; FONT-F
AMILY: Verdana, Tahoma, Helvetica, sans-serif; BACKGROUND-COLOR: #ffffff"
id="SB-191027">
<td id="SB-191028">Mandatory script</td>
<td id="SB-191029"><p style="font-family : Courier New; text-align : left;
border : thin groove" id="SB-191030">$ReplaceChars($ReplaceChars($(Mandato
ryScript.Source),"&", "&"), "<", "<")</p id="SB-191031"></td>
</tr>
$endif
$if $(DefaultScript.Source)< id="SB-191032">""
<tr style="PADDING-RIGHT: 2px; PADDING-LEFT: 2px; FONT-SIZE: 8pt; PADDING-
BOTTOM: 2px; VERTICAL-ALIGN: top; COLOR: #000066; PADDING-TOP: 2px; FONT-F
AMILY: Verdana, Tahoma, Helvetica, sans-serif; BACKGROUND-COLOR: #ffffff"
id="SB-191033">
<td id="SB-191034">Default value script</td>
<td id="SB-191035"><p style="font-family : Courier New; text-align : left;
border : thin groove" id="SB-191036">$ReplaceChars($ReplaceChars($(Default
Script.Source),"&", "&"), "<", "<")</p id="SB-191037"></td>
</tr>
$endif
$if ($(RelevantScript.Source)< id="SB-191038">""

```

```

<tr style="PADDING-RIGHT: 2px; PADDING-LEFT: 2px; FONT-SIZE: 8pt; PADDING-
BOTTOM: 2px; VERTICAL-ALIGN: top; COLOR: #000066; PADDING-TOP: 2px; FONT-F
AMILY: Verdana, Tahoma, Helvetica, sans-serif; BACKGROUND-COLOR: #ffffff"
id="SB-191039">
<td id="SB-191040">Relevance script</td>
<td id="SB-191041"><p style="font-family : Courier New; text-align : left;
border : thin groove" id="SB-191042">$ReplaceChars($ReplaceChars($(Relevan
tScript.Source),"&", "&"), "<", "<")</p id="SB-191043"></td>
</tr>
$endif
</table>
$endif
$endfor

<h1 style="FONT-WEIGHT: bold; FONT-SIZE: 18pt; COLOR: #000066; FONT-FAMILY
: Verdana" id="SB-191044">Scripts on links</h1>
$TableSQLName=$(SqlName)
$TableLabel=$(Label)
$for Links sort (SqlName ASC)
$if ($(RelevantScript.CalcMode) = 2)
<h2 style="FONT-WEIGHT: bold; FONT-SIZE: 10pt; COLOR: #000066; FONT-FAMILY
: Verdana" id="SB-191045">Link $(SqlName) ($(Label))</h2>
<table style="BORDER-RIGHT: #000066 1px solid; BORDER-TOP: #000066 1px sol
id; MARGIN-BOTTOM: 10px; BORDER-LEFT: #000066 1px solid; WIDTH: 400px; BOR
DER-BOTTOM: #000066 1px solid; table-width: 400px" id="SB-191046">
<tr style="PADDING-RIGHT: 2px; PADDING-LEFT: 2px; FONT-WEIGHT: bold; FONT-
SIZE: 8pt; PADDING-BOTTOM: 2px; COLOR: #ffffff; PADDING-TOP: 2px; FONT-FAM
ILY: Verdana, Helvetica, sans-serif; BACKGROUND-COLOR: #000066" id="SB-191
047">
<th id="SB-191048">Property</th>
<th id="SB-191049">Value</th>
</tr>

<tr style="PADDING-RIGHT: 2px; PADDING-LEFT: 2px; FONT-SIZE: 8pt; PADDING-
BOTTOM: 2px; VERTICAL-ALIGN: top; COLOR: #000066; PADDING-TOP: 2px; FONT-F
AMILY: Verdana, Tahoma, Helvetica, sans-serif; BACKGROUND-COLOR: #ffffff"
id="SB-191050">
<td id="SB-191051">SQL name</td>
<td id="SB-191052">$(SqlName)</td>
</tr>
<tr style="PADDING-RIGHT: 2px; PADDING-LEFT: 2px; FONT-SIZE: 8pt; PADDING-
BOTTOM: 2px; VERTICAL-ALIGN: top; COLOR: #000066; PADDING-TOP: 2px; FONT-F
AMILY: Verdana, Tahoma, Helvetica, sans-serif; BACKGROUND-COLOR: #ffffff"
id="SB-191053">
<td id="SB-191054">Name</td>
<td id="SB-191055">$(Label)</td>
</tr>
$if ($(RelevantScript.Source)< id="SB-191056">")
<tr style="PADDING-RIGHT: 2px; PADDING-LEFT: 2px; FONT-SIZE: 8pt; PADDING-
BOTTOM: 2px; VERTICAL-ALIGN: top; COLOR: #000066; PADDING-TOP: 2px; FONT-F
AMILY: Verdana, Tahoma, Helvetica, sans-serif; BACKGROUND-COLOR: #ffffff"
id="SB-191057">
<td id="SB-191058">Relevance script</td>
<td id="SB-191059"><p style="font-family : Courier New; text-align : left;
border : thin groove" id="SB-191060">$ReplaceChars($ReplaceChars($(Relevan
tScript.Source),"&", "&"), "<", "<")</p id="SB-191061"></td>

```

```

</tr>
$endif
</table>
$endif
$endfor
$endfor

$script
'-----
' Format a script to put it in a cfg
'-----
Function ScriptFormat(strMemos as String, iSpace as Integer) as String
ScriptFormat = ReplaceChars(strMemos, Chr(10), Chr(10) + Space(iSpace))
End Function

'-----
' Replaces a string with another one
'-----
Function ReplaceChars(strMemos as String, strToRep as String, strReplacement as String) as String
Dim I as Integer
ReplaceChars = strMemos
I = InStr(0, ReplaceChars, strToRep)
While (I < id="SB-191062"> 0)
ReplaceChars = Left(ReplaceChars, I - 1) + strReplacement + Mid(ReplaceChars, I + Len(strToRep), Len(ReplaceChars))
I = InStr(I + Len(strToRep), ReplaceChars, strToRep)
Wend
End Function
$endscript

```




B Determining the workflows used for a table

APPENDIX

This appendix aims to help you determine which workflows concern a given table in your AssetCenter implementation.

Workflows have a general context, also called the context of the start object. It is the table which is monitored for an event. The event can be a record inserted/deleted or a field updated, etc.

This context can change as the workflow progresses. Thus, each workflow activity can have its own context, different from the start context.

When searching for workflows operating on a given table, we can thus take the two following cases into account:

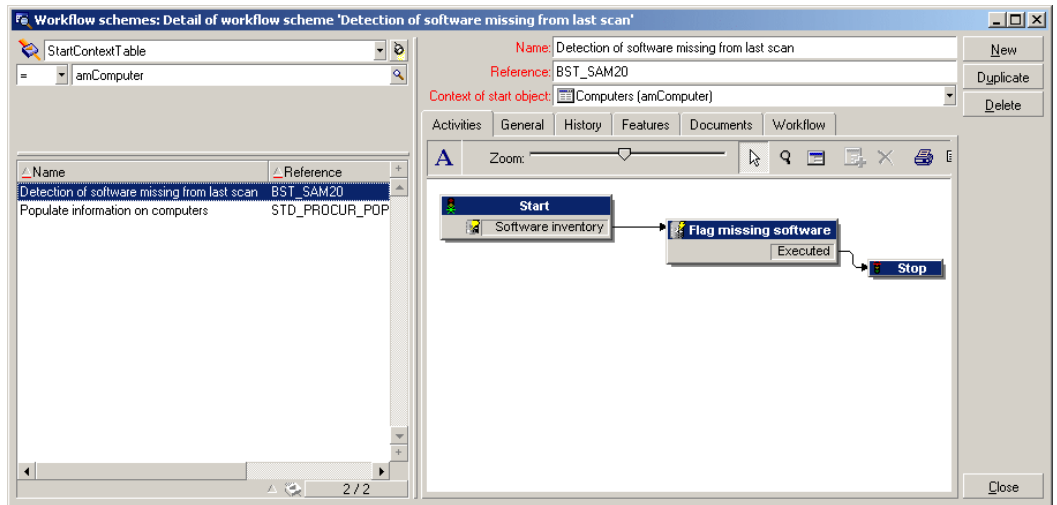
- The workflows whose start context is the table in question,
- The workflows with activities whose context is table in question.

In the following example, we are going to list all workflows concerning the Computers table (**amComputer**).

First, look for the workflows whose start context is the Computers table. To do this:

- 1 Start AssetCenter if it is not already running and then select **Tools/ Workflows/ Workflow schemes**.
- 2 Create a simple filter as shown below. Only those workflows whose start context is the **amComputer** table are displayed in the list. The list of workflows is as follows:
 - Detection of software missing from last scan

- Populate information on computers



Let's now look for workflows with one or more activities whose context is the Computers table. To do this:

- 1 Start AssetCenter if it not already running and then select **Administration/ List of screens**.
- 2 Select the **Workflow activities (sysamWfActivity)** screen from the list. AssetCenter displays the list of all the workflow activities.
- 3 Create a simple filter as shown below. Only those activities whose context is the **amComputer** table are displayed in the list with their associated workflow names. These are the following workflows:
 - Detection of software missing from last scan
 - Populate information on computers

- Trigger periodic inventory

The screenshot shows a software interface window titled "Workflow activities: Detail of workflow activity 'Computer inventory'". The window is divided into several sections:

- Table:** A table with columns for Name, Reference, Type, and Workflow. It lists three activities:

Name	Reference	Type	Workflow
Computer inventory	A001001	Automatic action	Trigger periodic inventory
Flag missing software	BST_SAM20-A001	Automatic action	Detection of software missing from last scan
Populate information on computers	POPUL-A02	Automatic action	Populate information on computers
- Form Fields:**
 - Name: Computer inventory
 - Template: [Dropdown]
 - Type: Automatic action
 - Reference: A001001
- General Tab:**
 - Log task:
 - Context: Computers (amComputer)
 - Workflow: Trigger periodic inventory
 - Input condition: OR
 - Comment: [Text area]

 **Note:**

The two workflows found earlier are of course included in this list since they have an activity (start activity) whose context matches our filter.



APPENDIX C Extracting the list of fields and links of the screens

This appendix aims to help you extract the list of fields and links of a screen of a given table.

AssetCenter Database Administrator, shipped with AssetCenter provides a template-based method to extract information (. **tpl** extension files).

This appendix includes a simplified and commented template that just extracts the list of the objects of the screens defined for the tables. This resulting list uses the pipe character "|" as a separator; You can change this by modifying the template. You can copy the contents to a file with the . **tpl** extension and execute it in AssetCenter Database Administrator.

 Note:

For further information on templates, refer to the **Administration** guide, chapter **Standard database description files**.

Executing a template in AssetCenter Database Administrator

To execute a template in AssetCenter Database Administrator, use the following procedure:

- 1 Start AssetCenter Database Administrator if it is not already running and connect to your database,
- 2 Select **Action/ Templates/ Select folder** and select the folder containing the template or templates you wish to execute,
- 3 Select **Action/ Templates/ Refresh list**. The list of available templates is displayed in the second section of the **Action/ Templates** menu.
- 4 Execute the script of your choice by selecting **Action/ Templates**, and then the name of the script.

Template example

```

$ Desc: Helper template (Tables - Screen - Fields)
$ Type: TXT
$ (c) Peregrine Systems, Inc. Documentation Team - 2004
$ Maintainer: Stéphane Bline
$ Warning: Do not modify this file directly. Send a formal change request
to me.

$ Specify the output folder for the list. A folder named fieldlist is crea
ted to store the result of the template execution
$OutputDir = $(Output.Path)
$MkDir$(OutputDir) + "fieldlist"
$ The output will be dumped to a text file name fields.txt
$SetOutput($(OutputDir) + "\fieldlist\fields.txt")
$ A first line containing the column titles is created
Table|Table Label|Field|Field Label|Screen|Screen Name|Tab|Tab Label
$ The template iterates on the screens defined within the database. For ea
ch one, the screen SQL name is retrieved
$for Screens sort (SqlName ASC)
$ScreensSQLName=$(SqlName)
$ The SQL Name and the label of the table attached to this screen is also
retrieved
$TableSQLName=$(Table.SqlName)
$TableLabel=$(Table.Label)
$ Now that the context is the screen, the script iterates on the tabs cont
ained in the screen and retrieves the tab SQL Name and label

```

```
$for Pages sort (SqlName ASC)
$PageSQLName=${SqlName}
$PageLabel=${Label}
$ If tab label is empty, then we are not inside a tab and the tab label and
SQL names are not meaningful anymore
$if ($(PageLabel)="")
$PageLabel="N/A"
$PageSQLName="N/A"
$endif
$ Now that the context is the tab, the script iterates on the elements con-
tained in this tab (fields, links, ...)
$ The script also retrieves the SQL Name and label of the object
$for Fields sort (SqlName ASC)
$FieldSQLName=${SqlName}
$FieldLabel=${Label}
$ For the sake of the example we are going to limit the output to a list of
fields and links.
$ If the Islink or Isfield conditional block below is removed then ALL obj-
ects will be retrieved (features, screen geometry, calculated fields,...)
$if $(IsLink) or $(IsField)
$ A line containing all the information is sent to the output file
$(TableSQLName) |$(TableLabel) |$(FieldSQLName) |$(FieldLabel) |$(ScreensSQLName)
|$(PageSQLName) |$(PageLabel)
$endif
$endif
$endfor
$endfor
$endfor

$script
```




Index

A

- Add NT users to the database (module), 24
- Add the computers listed in the NT domain to the database (module), 24
- AssetCenter Server
 - Overview, 23

C

- Calculate rents (module), 25
- Calculate stipulated loss values (module), 27
- Create assets, consumables, etc. corresponding to items received (module), 27

E

- Execute workflow rules for execution group (module), 28

L

- Let AssetCenter Server create the items received in the portfolio (option), 27

P

- Purge the input-events table (module), 30
- Purge the outgoing-events table (module), 30

S

- Search for new workflow execution groups (module), 31
- Signal presence of database server (module), 31
- Split expense lines in cost centers (module), 32

U

- Update statistics for tables (module), 30
- Update the database with the results of the scanners (module), 30 , 29

V

- Verify alarms (module), 34
- Verify history lines (module), 36
- Verify null-identifier records (module), 35
- Verify stocks (module), 36
- Verify time zone of database server (module), 33

