Product Documentation

Peregrine

Connect-It

AssetCenter Database Integration Solution

Part No. DCIT-360IP1-EN24

Peregrine SYSTEMS®

Build number: 31

Legal Notices

Warranty

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

Nothing herein should be construed as constituting an additional warranty.

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software.

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

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

Copyrights

Brands

- Adobe®, Adobe Photoshop® and Acrobat® are trademarks of Adobe Systems Incorporated.
- Corel® and Corel logo® are trademarks or registered trademarks of Corel Corporation or Corel Corporation Limited.
- Java™ is a US trademark of Sun Microsystems, Inc.
- Linux is a U.S. registered trademark of Linus Torvalds
- Microsoft®, Windows®, Windows NT® and Windows® XP are U.S. registered trademarks of Microsoft Corporation.
- Oracle® is a registered US trademark of Oracle Corporation, Redwood City, California.
- UNIX® is a registered trademark of The Open Group.

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

PEREGRINE

Table of Contents

PEREGRINE

Introduction

Who is this guide intended for?

This guide is aimed at anyone who needs to perform an integration to AssetCenter

How to use this guide

Chapter AssetCenter Software - General concepts

This chapter provides a general overview of AssetCenter, and explains the relationships that exist between natures, models, assets and portfolio items.

Chapter Mapping for an AssetCenter database

This chapter provides the main approach used for an AssetCenter database mapping, and then a more specific approach used for a computer mapping.

Chapter Mapping - Example

This chapter provides a detailed study of an Enterprise Desktop Discovery to AssetCenter database mapping.

Conventions used in this guide

Below is the list of conventions used in this guide:

Convention	Description
`JavaScript Code`	Example of code or command
`Fixed width characters`	DOS command, function parameter or data formatting.
...	Code or command omitted.
Note: Additional information...	Informative note
Important: Be careful...	Important information for the user
Tip: User tip	Tip
Warning: Warning	Extremely important information for the user
Object	Connect-It interface object: Menu, menu entry, tab or button.

The following conventions are also used:

- Steps to perform in a given order are presented in a numbered list. For example:
  1. First step
  2. Second step
  3. Third and last step
- All the figures and the tables are numbered according to the chapter in which they are found, and the order in which they appear in the chapter. For example, the title of the fourth table in chapter two will be prefixed by **Tableau 2-4**.

CHAPTER 1 AssetCenter Software - General concepts

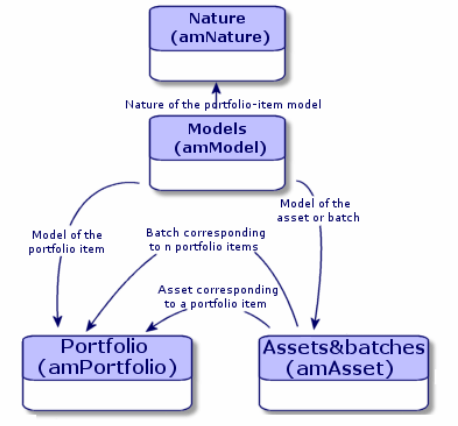
Implementing a Connect-It mapping requires detailed knowledge of AssetCenter's database structure, how portfolio items are created, and their relationships with other elements in the database such as employees, contracts or software.

- AssetCenter Guide - Concepts.
- AssetCenter Guide - Core tables.
- AssetCenter Guide - User interface, Welcome to AssetCenter, Introduction to the databases.
- AssetCenter Guide - Portfolio and Software licenses, Overview.

Portfolio item interdependencies

When creating items in the portfolio, notably a computer, it is important to follow the various creation steps and constraints.

As a reminder, the organization of the portfolio is based on models. Each model is based on a nature. Natures must be created prior to creating models.



Each nature specifies:

- In which table the models that are linked to this nature allow a record to be created.

  Example: A **Computer** nature is used to create models that are used to create computers in the portfolio items table. For natures that are used to create portfolio items, a second criterion - management constraint must be entered.
- Options for behavior can be selected for each nature that is used to create portfolio item models.

  Example: For a **Computer** nature, enabling the behavior option **Can be connected** will display tabs related to the connection ports.

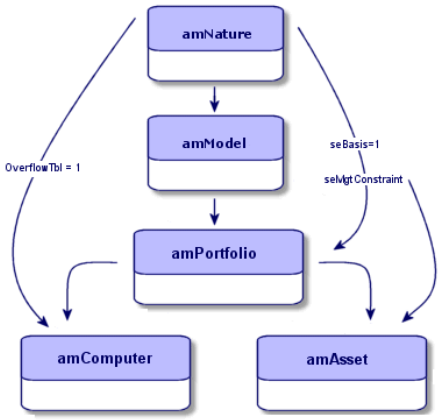
▸ AssetCenter Guide - Portfolio and Software licenses, Overview, Nature:. Creation and behavior.

Integrity rules

Follow the steps below to create a portfolio item:

- Define a nature whose behavior is **Create a portfolio item**
- Define a model based on a nature whose behavior is to create a portfolio item.

- Define a portfolio item based on a model whose nature creates a portfolio item.



▸ AssetCenter Guide - Physical data models.

Management constraints

A management constraint placed on a portfolio item (defined in the Natures table) must be taken into account in the Connect-It mapping.

Management constraints are as follows:

- Unique asset tag: Portfolio items that have their own asset tag are items that are tracked individually. Asset tags are used for the most important portfolio items that require financial monitoring. For example, a server.
- Unique asset tag: Portfolio items that share the same asset tag are grouped in a batch and are tracked collectively. Items in the same batch share the same asset tag. This management mode is recommended for identical items that do not need to be tracked individually. For example, a batch of 50 screens.
- Free: If a free management constraint is selected as the nature of a portfolio item, you can choose whether or not to associate an asset tag with it. Portfolio items without asset tags are those for which accurate tracking is not required. These items are grouped together in untracked batches and do not appear in the Assets table. For example, consumables whose tracking is done indirectly via items that use the consumables.

For a management constraint whose type is:

- Unique asset tag, asset tag, shared asset tag, an item created will be created in the portfolio items table and in the assets table.
- Free, an item created is only recorded in the portfolio items table.

▶ AssetCenter Guide - Portfolio and Software licenses, Overview.

Overflow tables

Certain portfolio items require specific fields. These fields are stored in special tables called overflow tables.

Each time one or more overflow tables are specified for a portfolio item record, this record is simultaneously created in the portfolio items table and in the overflow tables: the Assets table and the Computers table, for example. Each time a record is created or deleted from one of these tables, the same is automatically done in the other tables.

Overflow tables facilitate the integration of other applications into AssetCenter. For example, AssetCenter can integrate network inventory information and present this information in the Computers overflow table.

The main AssetCenter overflow tables are as follows:

- Assets table
- Computers table
- Software installations table
- Monitors table

As a result of this data model, a link to the portfolio items table and to the Assets table exists for an amComputer document.

▶ AssetCenter Guide - Portfolio and Software licenses, Overview, Overflow tables.

CHAPTER 2 Mapping for an AssetCenter database

A Connect-It mapping for an AssetCenter database uses the same overall structure similar to the other demonstration scenarios supplied with the application.

Customizing the Asset Management connector

The Asset Management connector has been specially developed for use with an AssetCenter database.

When the connector is used and the scenario file opened, the configuration file is loaded in the following order:

1. Default configurations of the Connect-It installation folder **config\shared**
2. Default configurations of the Connect-It installation folder **config\ac**
3. User configurations of the Connect-It installation folder **scenario\[scenario name]**

Note:

The function name must be unique. An error message is displayed if two functions have the same name.

▸ Connect-It - Connector Guide, Peregrine Systems Connectors, Asset Management Connector.

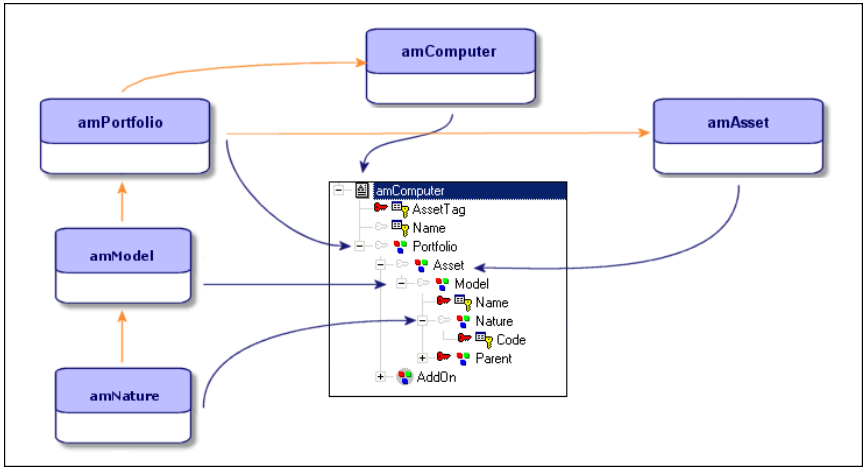
Language used

To be language independent, save common strings that will be called and used in the Connect-It scripts to be inserted as is in a target database. These strings are saved in a **`.str`** file and called using the **PifStrVal** function.

Main approach

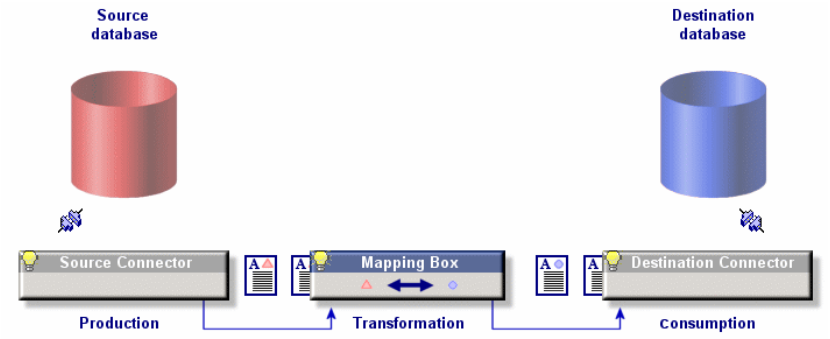
The document type produced or consumed by the Asset Management connector is an interpretation of the structure of an AssetCenter database schema.

Consequently, the mapping must reflect the item's creation constraints.



Mapping - Overview

A mapping is created when elements of a source document and elements of a destination document are linked.



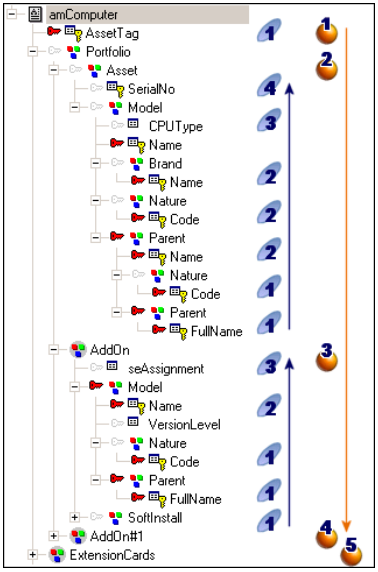
Processing a scenario

When a scenario is executed, the following operations are performed:

1. Creating the document type produced by the source connector
2. The document type from the mapping is created
3. Reconciliation is performed for the destination connector: The document's elements from the mapping are compared with the fields present for a given record in the database
4. Reconciliation scripts are applied for the destination connector

▶ Connect-It - User's Guide, Implementing an integration scenario, Defining mappings of document types.

Processing a mapping



The processing order of complex elements that make up a mapping is as follows:

1. First level elements composing a document type: fields (1)
2. Second level elements composing a document type: structures (2)

   Structures are processed before collections. If a collection is present within a structure, it will be processed only after all of the structures present within the entire document type are processed (1, 2, 3, 4, 5.)
3. Third level elements composing a document type: collections (3)

Important:

This processing order must be implemented when using the **Use the parent ID as a reconciliation key** option. This option is used to follow the link that connects two portfolio items and use the reconciliation key of the parent item. This option has the opposite behavior of the **Follow the link** option.

▸ Connect-It - Connector Guide, Connector directives.

AssetCenter management constraints

Depending on the management constraints set for an item, the inventory mapping and the data to migrate will point to different tables.

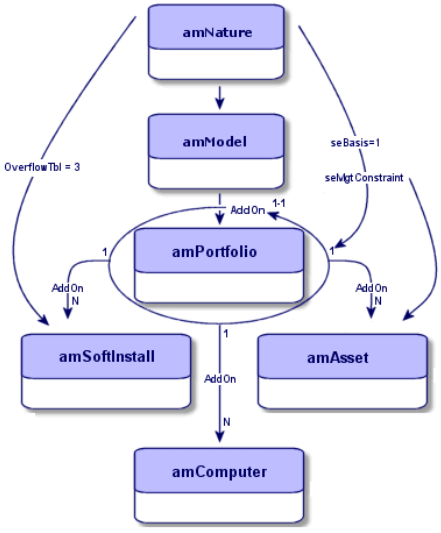
A mapping migrating information related to:

- Computers, will use the amComputer document type
- Software will use the amSoftInstall document type

Note:

Software installations associated with a computer use a special processing method and are represented by an AddOn collection that is visible in the amComputer document type.

It is still possible, within the same document type, to manage all information using the link that joins the different AssetCenter tables. For example, you can move information related to departments and employees via the link that is available in the portfolio items table.



For each of these document types, a link to the complex Portfolio item is created because of the management constraint that is applied (a nature which creates

a portfolio item by default and additional information have been entered in the linked overflow tables).

▸ Connect-It - User's Guide, Implementing an integration scenario, Defining mappings of document types.

▸ AssetCenter Guide - Administration, Standard database description files.

Preparing a mapping to an AssetCenter database

Before determining key elements for a mapping, you must:

- View the document type used by the AssetCenter connector and delete information that is not needed, such as complex elements (fields, structures, collections) that will not be used in AssetCenter.
- Identify important information (structures, collections) of the document type that is produced or consumed for the mapping (identify the fields and links of the tables that will be used in AssetCenter).

Determining key elements for a mapping

Because of integrity rules used by the AssetCenter database model, the key elements for a mapping using an Asset Management connector are dependent on what the mapping is required to do. Consequently, the mandatory fields to complete may change.

Also, the line-of-business involved as well as client database customization should be taken into account. For example, if a line-of-business rule requires that an element assigned to a user go through a stock, the mapping must take this into account to avoid assigning an item directly to a user.

Creating an item AssetCenter in implies:

- Respecting integrity rules specific to AssetCenter
- Identifying items that make up the reconciliation key
- Identifying the item to insert depending on the information that was retrieved

Following the portfolio item creation prerequisites as previously explained, the following mappings describe the main items that are required to create a record in AssetCenter.

▸ Connect-It - User's Guide, Implementing an integration scenario, Defining produced or consumed document types.

Creating a nature

In AssetCenter, a nature that is created requires the following information:

- amNature root document: Name, Code elements

  It is recommended to use the nature's code rather than the nature's name as it may change depending on the installed language version.

If no value is defined in the mapping, these fields are automatically completed via a script in AssetCenter.

Important:

It is not recommended to use Connect-It to create a nature. Use AssetCenter instead.

However, if natures must be created in the mapping, you will need to use all fields required for creating the nature in the mapping.

By default, a nature creates a portfolio item with the unique asset tag as management constraint.

▸ AssetCenter Guide Portfolio and Software licenses, Portfolio items, Natures.

Creating a model

In AssetCenter, a model that is created requires a nature.

For an amModel document type, the following items are required when creating a model:

- amModel root document: Name, Barcode elements

Important:

The name of a model is not unique, only its FullName is unique.

- Nature structure: Name, Code elements

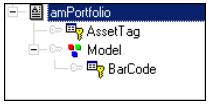
Only the elements listed above are mandatory. If no other value is specified in the mapping, the **Barcode** field automatically receives a value.

Note:

You can, however, populate an AssetCenter database with pertinent information using additional, optional fields.

▸ AssetCenter Guide Portfolio and Software licenses, Portfolio items, Models.

Creating a portfolio item



In AssetCenter, a portfolio item that is created requires the following information:

- amPortfolio root document:
- Model structure: Barcodes

Only the elements listed above are mandatory. If no other value is specified in the mapping, the **Barcode** field automatically receives a value.

Note:

Using the **Follow the link** option enables you not to specify a first-level element and to select a second-level element (for example, an asset's serial number) as the reconciliation key.

▸ Connect-It Guide - Peregrine Systems Connectors, Asset Management Connector, Production directives of the Asset Management connector.

Determining a reconciliation key

Determining how a reconciliation key is selected depends on how your company has decided to manage its assets. If, for example, each asset is monitored individually in the procurement cycle, then selecting the computer's identifier or its serial number as the reconciliation key seems evident.

However, if the procurement cycle is not monitored in AssetCenter, then an element that remains constant over time must be defined as the reconciliation key. This may be the computer's network name (**Name** for the **amComputer** document type) or its MAC address.

Determining a reconciliation key consists in determining:

- One or more fields available in the source database and present in the AssetCenter database
- One or more solid reconciliation keys, meaning a field whose value remains constant during the item's life cycle

Tip:

Reconciliation is often done using the element's FullName in order to ensure reconciliation of the correct element. It is recommended to save the element's FullName in a **`.str`** file and to call it using the **PifStrVal** function. This reduces the chance of having to modify the entire mapping if the element's FullName is changed.

▸ Connect-It User's Guide, Implementing an integration scenario, Defining mappings of document types, Reconciliation keys.

Behavior of a reconciliation key

When a reconciliation key is defined, the field containing the key is used to uniquely identify the document produced in order to compare it with a given record in the destination table.

Reconciliation key on a link

When a reconciliation key is set on a link (a structure) a new value is inserted instead of replacing the old one.

Determining alternate reconciliation keys

If a failure occurs on the first key, Connect-It allows you to define one or more alternate reconciliation keys.

Determining alternate reconciliation keys has two main objectives:

- Search and define valid values for a given period that are not fixed over time, and that enable the primary key to be propagated (for example, if a unique identifier does not exist for a given computer, an alternate reconciliation solution can be implemented using the computer's name or MAC address).
- Search and define, for several fields containing fixed keys in the source, a specific key that depends on the portfolio item's type which is being processed by an identical mapping (router, computer)

▸ Enterprise Discovery to AssetCenter scenario [page 31].

▸ Connect-It - User's Guide, Implementing an integration scenario, Editing scenario options, Display.

Mapping script associated with a reconciliation key

A mapping script can be associated with a reconciliation key in order to define its behavior when it encounters a value.

Create a mapping script to assign a value on the fly to an element that is used as the reconciliation key.

Two cases are possible:

- The value to be inserted is unique
- The value to be inserted can be an existing value (a model or a nature).

Using a reconciliation key implies the following:

- For each field that is used as a reconciliation key, the selected element has a value in AssetCenter.

  If no value is available, the reconciliation cannot be performed and an error is saved in the document log.

  Note:

  A null value (empty string) can be used.

- Inserting a null value can violate an integrity rule (for example, an asset's asset tag must always have a value)

Tip:

Use the **Show queries in tracking lines** option to improve visibility of queries sent to the AssetCenter database.

▸ Connect-It - User's Guide, Implementing an integration scenario, Defining monitors.

▸ Connect-It - User's Guide, Implementing an integration scenario, Editing scenario options, Connectors.

Missing source value

To insert a unique value in the destination database when no value is available in the source database, you can:

- Create a mapping script
- Use a global function

It is recommended to use a global function as it allows you to share the same script at different locations in the mapping and improves readability.

For example, a global function can create an identifier by concatenating the machine's network identification information (name, group and domain to which it belongs).

▸ Connect-It - User's Guide, Implementing an integration scenario, Defining mapping scripts, Editing associated files.

Defining a unique value for a reconciliation key

A default value can be defined in a mapping script.

For example, an inventory tool sends information related to a computer, but no model is associated with this computer in the AssetCenter database: The script defines that a default value be assigned and the model Unknown be used if the value is empty.

▸ Connect-It - User's Guide, Implementing an integration scenario, Defining mapping scripts.

Using a reconciliation script

A reconciliation script is applied after a mapping script. It is used to perform the following actions for a document type consumed by the Asset Management connector:

- Update if the value exists in the target database
- Insert if the value is missing from the target database

Using a reconciliation script results in the following:

1. The item of the list of fields to update for the record is deleted.
2. - In update mode, the existing value is maintained.
   - In creation mode, the default value is used.

Using reconciliation scripts in this manner is valid when reconciliation is done for an item that has a reconciliation key, such as the AssetTag, which cannot have a null value.

▸ Connect-It - Connector Guide, Connector directives, Reconciliation.

The reconciliation cache

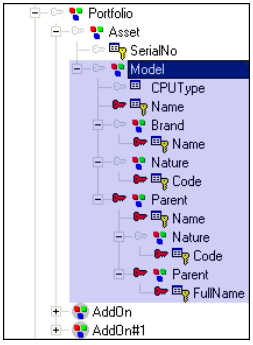
The reconciliation cache is used to reduce execution times and the number of database queries. The cache is stored in memory.

The reconciliation cache is useful when:

- constants are used in a mapping
- when the mapping concerns relatively small tables such as the Models (amModel) or Natures (amNature) tables
- when the mapping concerns tables that will not be updated

▸ Connect-It - Connector Guide, Peregrine Systems Connectors, Asset Management Connector.

The screen shot below shows the set of structures concerned by the reconciliation cache used during an inventory scenario.



The reconciliation cache optimizes scenario performance for a given session by reducing the number of queries.

The reconciliation cache operates in the following manner:

1. The Connect-It session is opened
2. The values of the reconciliation keys are stored in memory.
3. The unique identifier corresponding to the reconciliation keys are stored in the cache

4 For each cached element used in the mapping, no query is sent

Note:

Once the item or document is stored in the cache, the value is acquired and is no longer up-to-date for the current session.

5 Purge of the cache when the maximum number of documents defined has been reached.

6 The cache is purged at the end of the session

The maximum number of document stored in the cache is defined in the options that are associated with the connectors (Edit/Options/Connector).

Parallelization

Parallelization is applied to all connectors consuming a document type. It consists of duplicating a connector into multiple processes in order to process document consumption in parallel.

Performance improvements linked to parallelization depend on the difference between the speed of document production and the speed of document consumption. For example, if a scenario produces documents more slowly than it consumes them, parallelizing consumption will be of no benefit. Other factors can influence performance, such as database architecture, network issues, etc.

Deadlocks

Deadlocks linked to writing to a database may occur when two processes try to access the same record. The first process accesses the record and blocks the second process from accessing it.

Different methods can be implemented to avoid deadlock situations:

- In Connect-It, limit the number of documents per transaction (<100).
- In Connect-It, avoid concurrent accesses in the same table, for example, the models table.
- In Connect-It, split a scenario into two mappings where the first one creates the repository (insertion of models, employees, locations or natures) and does not work in parallelization mode. The second mapping reads the data from the repository and inserts the new data in parallelization mode.
- In Connect-It use the cache to avoid write access

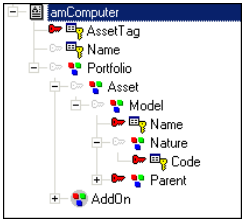
- In AssetCenter, avoid using counters, notably in the AssetCenter wizards, and moving them to the overflow tables.
- Keep in mind the specifics of the DBMS being used (DB2, Oracle, etc.).

▶ AssetCenter Guide - Tuning, Tuning the database, Eliminating locks and deadlocks.

Specific approach Mapping a computer

Creating a computer

To create a computer, the mapping structure is the following:



In this mapping type, a computer is linked to an asset and to a portfolio item, which is itself dependent on a model.

The elements that are required when creating a portfolio item are the following:

1. Root document amComputer: AssetTag elements, Name
2. Portfolio structure
3. Asset structure
4. Model structure

- Portfolio structure

  A portfolio item can be linked to a record in an overflow table. This behavior is defined in the model's nature that creates the portfolio item. Consequently, a portfolio item will be linked to the computers table and to the assets table (which depends on its management constraint).

Important:

The nature of the model on which the asset is based must be **Computer**.

- Asset structure

  This structure is only required if it includes an element that is required when creating or updating a portfolio item such as an asset's serial number or a link to a model.

  When a nature entails the creation of a portfolio item, it also allows a linked creation to be defined (computer, monitor, software installation, telephone). A unique or shared asset tag type management constraint is associated with this creation that will create a linked record in the assets table at the same time the portfolio item is created. Once it is created, an asset's nature cannot be changed.

  The assets table also contains the AssetTag of a portfolio item. If it is used in a data model, then you must use this table when mapping portfolio items.
- Model structure

  The creation of a record in the assets table is done based on a model, just like the creation of a portfolio item. For this reason, the model used to create a record is included in the mapping.

Selecting required keys when creating a computer

For mapping elements that are required when creating or updating a Computer type record in AssetCenter, the reconciliation keys correspond to the following elements:

- **AssetTag** element:

  The identifier (**Asset Tag** field) of a computer is a constant. This value should remain constant over time and allow a computer to be identified uniquely. Defining a reconciliation key for this item will enable the item to be based on a constant value.
- **Name** element: If your portfolio management model is not based on a unique identifier for a computer, a barcode for example, then choose another element that will act as the reconciliation key. The computer name (**Name** field) is an alternative method to uniquely identify an item in the IT portfolio. This field

is automatically populated by an AssetCenter agent if no value is specified in the mapping script or if this element is not included in the mapping.

Important:

The name of the machine that is identified on the network must not change. If it does change, a new computer will be created when the inventory tool performs a new inventory of the network.

- **PhysicalAddress** and **TcpIpAddress** elements: These elements can be used as reconciliation key if no identifier or unique name is available.
- **Name** (Model.Name) element:

The name of a model is mandatory. As a portfolio item is linked to an asset which is itself linked to a model, reconciliation is done on the model's **Name** field.

To summarize:

- To uniquely identify a computer, the Asset Tag is used.
- To create a computer, the asset tag, the model name and the link between the portfolio items table and the models table are required.

Mapping scripts

Define a mapping script for each reconciliation key.

This mapping script can be defined:

- Directly via Connect-It when two elements are linked
- Manually via a Basic script

▸ Connect-It - User's Guide, Implementing an integration scenario, Defining mapping scripts.

Follow the link

This option, which can be accessed for any selected structure, is useful:

- For the overflow tables. Following the link for an item in a reference table lets you retrieve information from the linked table.

- When you cannot define a reliable reconciliation key (for example, if you cannot perform a reconciliation using the Employee ID but only with their Name and First Name).

▸ Connect-It - Connector Guide, Peregrine Systems Connectors, Asset Management Connector, Production directives of the Asset Management connector.

For mappings that create or update information related to a computer, the follow the link option is enabled for the following structures:

- Portfolio

  The amComputer table is an overflow table of the amPortfolio table. Therefore, it is unnecessary to perform queries to check the integrity linked to the amPortfolio table as it is linked to the amComputer table via a 1-1 link.

  If a computer exists then a portfolio item exists and no request is made.

  Note:

  Because of integrity rules specific to AssetCenter, when a computer is created, an agent is triggered and automatically creates the corresponding record in the portfolio items table.

- Asset

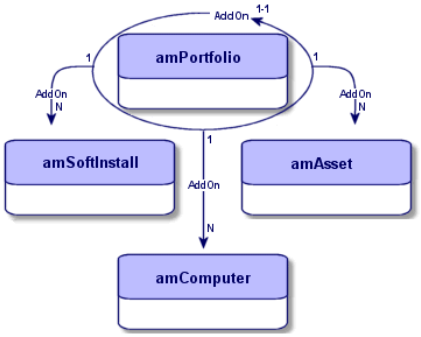
  The properties that are being searched for correspond to the models table. Only one model corresponds to a portfolio item.

  The properties that are being searched for correspond to the natures table. Only one nature corresponds to a model.

Information linked to a computer

When a record whose type is computer is created in the AssetCenter database, certain information linked to the computer are recorded in adjacent tables. This is the case for information related to network cards, extension cards or physical hard drives. This information can be viewed, for an amComputer document type mapping, in collections whose name corresponds to the name in the AssetCenter table or in AddOn type collections.

AddOn Collections



AddOn type collections that correspond to the **Component** (AddOn) link, that can be viewed in the portfolio items table (amPortfolio), OwnCopy link type, are used to:

- Define a tree-structure.
- Link a portfolio item to another portfolio item that is part of an overflow table.

AddOn collections group the following information.

- Information related to software installations: AddOn collection, SoftInstall structure
- Information related to monitors: AddOn#1 collection, Model structure.

Note:

The name of the AddOn collections (AddOn#1, AddOn#2, etc.) depends on the order in which the mapping was written.

These collections correspond to links defined in the **Component** tab for a record of the portfolio items table.

Using AddOn type collections depends on the management type that is defined by the user.

Use the parent ID as the reconciliation key option

This option is used by reconciliation in order to process only AddOn type elements that belong to the parent portfolio item.

ExtensionCards, LogicalDrives, NetworkCards, PhysicalDrives collections

Information linked to a computer and that is not part of the categories defined for AddOn collections are recorded in the following tables:

- amExtensionCards
- amLogicalDrives
- amNetworkCards
- am PhysicalDrives

These tables contain additional information about a computer but are not overflow tables of the portfolio items table (amPortfolio).

These tables correspond to the **Extension**, **Disks** and **Network** tabs that can be viewed in the computers table.

CHAPTER 3 Mapping - Example

This chapter describes in detail the mapping structure for the sample scenario, **edac.scn**, supplied for the Enterprise Discovery connector.

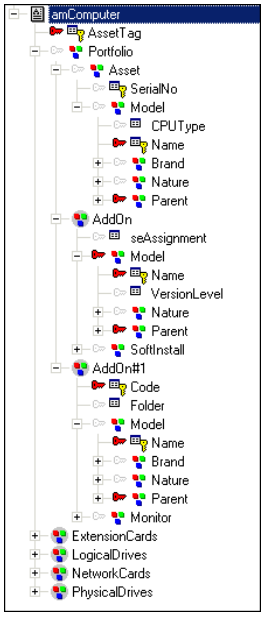
Enterprise Discovery to AssetCenter scenario

This scenario enables you to transfer data from an Enterprise Network Discovery database to an AssetCenter database.

The mapping described below is the Network-Devices **Structure-amPortfolioDst** mapping.

Mapping structure

The mapping is structured in the following manner:



This structure follows the schema described in the previous chapters.

Note that the mapping starts from the created item, the computer, and links to the elements that are used to create (portfolio item, model, nature). This is the opposite of the creation process used by AssetCenter, which calls all the elements (nature, model) that are required to create and insert an item.

▸ Processing a mapping [page 14].

The **edac.scn** mapping involves the **amComputer** document type and its dependencies.

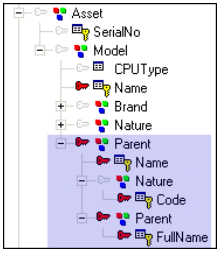
- Portfolio items
- models
- natures
- brands
- Software installations
- monitors
- Extension cards

- internal devices

A table in AssetCenter exists for each complex element present in the mapping.

- Portfolio: amPortfolio
- Model: amModel
- Nature: amNature
- Brand: amBrand
- AddOn.SoftInstall: amSoftInstall
- AddOn.Monitor: amMonitor
- NetworkCards: amNetworkCard
- LogicalDrives: amLogicalDrive
- PhysicalDrives: amPhysicalDrive

Parent structure



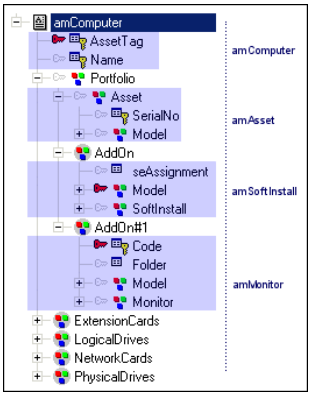
In AssetCenter several models can share the same name and have different natures. Since models can not be differentiated from others using just the name, you can differentiate them by identifying the model's parent or the nature associated with the model. For example, the EAI model may result in the creation of a software or software license installation depending on its nature.

AssetCenter uses constant values such as sysComputer or MODEL_WORKSTATION_AC44. These values are used in the mapping scripts. When an inventory returns data that is not present in the AssetCenter database, these values enable a computer to be inserted after assigning it a model that is linked to a nature which is used to create a computer.

Identifying reconciliation elements

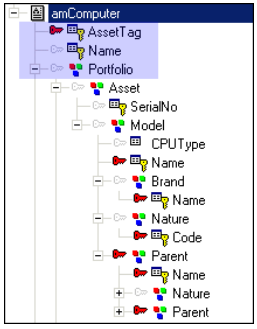
In a mapping, reconciliation elements that are required to create a computer must be distinguished from elements required to create information linked to

a computer. Generally, these elements (structures, collections) correspond to the table names visible in AssetCenter.



Because of interdependencies that exist between a computer, a portfolio item and an asset, elements that create the interdependency are found in the computer creation or update mapping.

It is important to distinguish the role of an element in a mapping: Using an element and assigning a reconciliation key to it does not imply that this reconciliation key plays the same role as another key within a mapping.



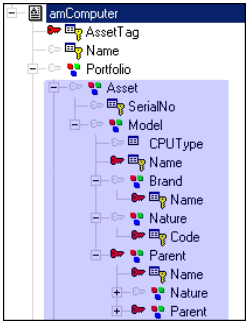
Thus, the key positioned on the complex element **AssetTag** is used to uniquely retrieve a computer depending on its identifier, but does not create it. Creating or updating a computer depends on its dependencies. In the same manner, the reconciliation key set on the complex element **Brand.Name**, implies the creation or update of this element, without checking the integrity of the computer information itself.

Choosing reconciliation keys

The problems to resolve for this scenario are the following:

- Determine how to create an identifier for a computer when the first network inventory is implemented and when no identifier exists for a given computer.
- Select reliable reconciliation keys that will last over time and that will allow reconciliation of values of scanned items with values saved in an AssetCenter database.

Reconciliation elements for this scenario follow the general structure as described in this guide:



For the Enterprise Discovery to AssetCenter scenario, the choice of reconciliation keys are the following:

- **AssetTag** element: For an inventory scenario, an identifier linked to an item of the IT portfolio allows for this item to be uniquely identified.

  Defining the AssetTag element as the primary reconciliation key implies that each record is identified by its AssetTag when the network inventory information is retrieved.
- **Name** (Model.Name) element: A portfolio item, a computer, that can only be linked to a single model. The name of the model is selected as the reconciliation key.
- **Name** (Brand.Name) element: The inventory tool retrieves information linked to the brand. Reconciliation is done using the brand's name to prevent adding a new brand after each inventory and to reconcile values with existing values in the AssetCenter database.
- **Code** (Nature.Code) element: A nature defines what a model creates (portfolio item, asset, etc.) The nature's code is used as reconciliation key as it uniquely identifies a nature.

- **Name** (Parent.Name) element: A model can be composed of other models, thus creating a hierarchy. The name of a parent model is used as the reconciliation key.
- **Code** (Nature.Code) element: A nature defines what a model creates (portfolio item, asset, etc.) The nature's code is used as reconciliation key as it uniquely identifies a nature.
- **FullName** (Parent.FullName) element: The FullName is what is used lastly to define where the model belongs on the hierarchy. The FullName is unique and is used as a reconciliation key.

Alternate reconciliation keys

If a failure occurs on the first key, it is possible to define alternate reconciliation keys for a given key set. Three key sets are defined for this scenario: A set of primary keys and two sets of alternate reconciliation keys.

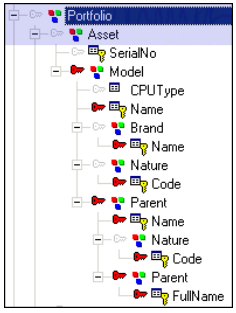
The keys that are defined are the following:

- **PhysicalAddress**: If there is a failure for the first reconciliation key (AssetTag), this element is defined as the second reconciliation key, if it has been determined that a computer can be identified on the network via its MAC address.
- **TcpIpAddress**: If there is a failure for the second reconciliation key (PhysicalAddress), this element is defined as the last reconciliation key, if it has been determined that an IP address is a stable element for a given network.

▸ Determining alternate reconciliation keys [page 19]

Selecting the Follow the link option

When items are linked together via a 1-1 link, you are not required to send a query to verify the integrity linked to the table. Enable the **Follow the link** option to avoid performing the query.



The **Follow the link** option is enabled for the structure:

- Portfolio
- Asset

Consequently, no reconciliation key is defined for the sub-items of these structures.

Mapping script associated with the reconciliation key

The script associated with the reconciliation key allows a value to be inserted or updated in the AssetCenter database following the same data format process.

- Script for the **AssetTag** element:

```
ToSmart(EDDIGetComputerModel ( [hwSMBIOS.hwSmbiosSystemInformation(0).hwsmbiosProductName], [hwBiosData.hwBiosMachineModel], [Model.Model_Name], [DeviceCategory.DeviceCategory_Description]))
```

This script calls the **EDDIGetComputerModel** function using the values of the following fields as parameters:

- hwSMBIOS.hwSmbiosSystemInformation(0).hwsmbiosProductName
- hwBiosData.hwBiosMachineModel
- Model.Model_Name
- DeviceCategory.DeviceCategory_Description

If no value exists for the first three parameters, the function returns the value of the last parameter.

Next, the value is formatted by the **ToSmart** instruction.

- Script for the **Name** (Model.Name) element:

```
UCase(EDDIGetACAssetTag ([hwAssetData.hwAssetTag], [hwNetworkData.hwNet
workNames.hwWorkgroupName], [hwNetworkData.hwNetworkNames.hwLocalMachin
eID], [NMID.Appliance.Appliance_ServerID], [NMID.NMID_NMID]))
```

  This script calls the **EDDIGetAssetTag** function using the values of the following fields as parameters:
  - hwAssetData.hwAssetTag
  - hwNetworkData.hwNetworkNames.hwWorkgroupName
  - hwNetworkData.hwNetworkNames.hwLocalMachineID
  - NMID.Appliance.Appliance_ServerID
  - NMID.NMID_NMID

  Next, the value is formatted by the **UCase** instruction.
- Script for the **Name** (Brand.Name) element:

```
Dim strBrand As String

strBrand = EDDIGetComputerManufacturer ([hwSMBIOS.hwSmbiosSystemInforma
tion(0).hwsmbiosSystemManufacturer], [CompanyHW.Company_Name])
If strBrand = "" Then
strBrand = PifStrVal("UNKNOWN")
End If

RetVal = ToSmart(strBrand)
```

  This script calls the **EDDIGetComputerManufacturer** function using the values of the following fields as parameters:
  - hwSMBIOS.hwSmbiosSystemInformation(0).hwsmbiosSystemManufacturer]
  - CompanyHW.Company_Name

  The name results from the concatenation done by the **EDDIGetComputerManufacturer** function. If no value is returned by the function, an UNKNOWN model is created. If a value is retrieved by the **EDDIGetComputerManufacturer** function, it is formatted by the **ToSmart** function.
- Script for the **Code** (Nature.Code) element:

```
"sysComputer"
```

  This value defines the code for the nature that is used during automatic creation of an asset associated with a computer via the computers table.
- Script for the **Name** (Parent.Name) element:

```
Dim strBrand As String

strBrand = EDDIGetComputerManufacturer ([hwSMBIOS.hwSmbiosSystemInforma
tion(0).hwsmbiosSystemManufacturer], [CompanyHW.Company_Name])
```

```
If strBrand = "" Then
strBrand = PifStrVal("UNKNOWN")
End If

RetVal = ToSmart(strBrand)
```

The name of the parent item results from the concatenation done by the **EDDIGetComputerManufacturer** function. If no value is returned by the function, an UNKNOWN model is created. If a value is retrieved by the **EDDIGetComputerManufacturer** function, it is formatted by the **ToSmart** function.

- Script for the **FullName** (Parent.FullName) element:

```
PifStrVal("MODEL_WORKSTATION_AC44")
```

A default value is defined for all sub-models using the identifier MODEL_WORKSTATION_AC44. This identifier enables the character string that corresponds to the local language to be used. Consequently, each sub-item of a given inventoried item is linked to a model by a parent-child link and takes the value of the MODEL_WORKSTATION_AC44 identifier.

▸ Connect-It - User's Guide, Implementing an integration scenario, Defining mapping scripts, Editing associated files.

▸ Connect-It Guide - Programmer's reference.

Global functions associated with the mapping

Several functions were developed for this mapping which are mainly used to send information for required AssetCenter fields if this information is missing during the inventory.

For example, the **EDDIGetNMID** function creates a unique identifier for a computer.

The description of functions given below is not exhaustive and is mainly linked to functions used to create an identifier on the fly.

- **EDDIGetNMID** function:

```
Function EDDIGetNMID(ByVal strServerID       As String, _
ByVal strNMID           As String) As String
EDDIGetNMID = strServerID & "." & strNMID
End Function
```

Two parameters are required for this function:

- strServerID
- strNMID

The parameters are concatenated to return a unique identifier.

- **EDDIGetACAssetTag** function:

```
Function EDDIGetACAssetTag (ByVal strAssetTag       As String, _
ByVal strWorkGroupName  As String, _
ByVal strLocalMachineID AS String, _
ByVal strServerID       As String, _
ByVal strNMID           As String) As String
If strAssetTag <> "" Then
EDDIGetACAssetTag = strAssetTag
ElseIf strWorkGroupName  <> "" _
AND strLocalMachineID <> "" Then
EDDIGetACAssetTag = strWorkGroupName & "_" & strLocalMachineID
Else
EDDIGetACAssetTag = EDDIGetSCLogicalName ( strServerID, strNMID )
End If
End Function
```

Several parameters are expected for this function. If the first four parameters do not return a value, the value of the last parameter, strNMID, is used.

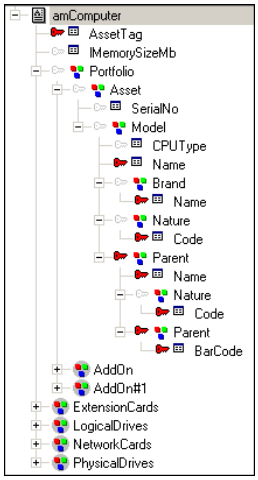
Enterprise Discovery to AssetCenter scenario - Reconciling inventory data

This scenario is used to reconcile inventory data for an AssetCenter application.

This scenario comprises several mappings which are used to create records in the computers table and linked tables: Portfolio items, models and natures.

The structure of the mapping of this scenario is similar to the **edac.scn** scenario.

The mapping described below is the **DevicesSrc-amComputerDst** mapping.



Note:

For AssetCenter version 4.4, you create a model whose FullName is **/Network/Network Device Component/**. This model is associated with the **Network hardware** nature (code **NET**).

The reconciliation proposal is done for the **lMemorySizeMb** element.

Reconciliation proposals

For this scenario, reconciliation proposals that are available in AssetCenter are created from the script for the **lMemorySizeMb** element.

The reconciliation script is as follows:

```
If vNewVal >= vOldVal Then
RetVal = vNewVal
Else
RetVal = ValidateReconcUpdate("CPU_MEM_" & [AssetTag] & [dtHardScan], Form
atResString(PifStrVal("RECONC_SAMPLE_LOWER_MEMORY"), [Name]), "amComputer"
, "lMemorySizeMb", vNewVal, vOldVal, vOldId)
End If
```

This script compares the value that is retrieved by the inventory with the value present in the AssetCenter database:

- If the new value is greater than the former value, it is inserted in the database.

- If the new value is less than the value in the database, then a reconciliation proposal is provided to the database administrator to allow him or her to validate whether or not the computer memory has been reduced. The status of the Connect-It document becomes **Pending**.

This script uses the **ValidateReconcUpdate** Basic function which, by calling a specially designed function, enables records pending validation to be created in the reconciliation proposals table.

This function checks if a reconciliation proposal already exists by verifying its identifier.

Note:

This function must be used with an update script (**Update script** frame).

- If no reconciliation proposal exists, a reconciliation proposal is created and the status of the record is set to **To assign**.
- If a reconciliation proposal already exists and its status is:
  - **Validated**, the new value for the element is inserted into the corresponding table.
  - **Rejected**, the document is not processed. No update is performed in AssetCenter.
  - For any other status, the status of the record is **To assign**.

```
'------------------------------------------------------------------------
---
' strCode  = unique identifier for the reconciliation proposal
' strName  = description of the reconciliation proposal
' strTable = table concerned by the change
' strField = field whose value has changed
' vNew     = new value of the field
' vOld     = old value of the field
' lRecId   = main ID of the modified field
'------------------------------------------------------------------------
---
Function ValidateReconcUpdate(ByVal strCode As String, ByVal strName As St
ring, ByVal strTable As String, ByVal strField As String, ByVal vNew As Va
riant, ByVal vOld As Variant, ByVal lRecId As Long) As Variant
ValidateReconcUpdate = CheckReconcProposal(strCode, strName, strTable, str
Field, vNew, vOld, lRecId)
End Function
```

When this scenario is replayed, Connect-It checks the status of the reconciliation proposal. If it has been validated by the database administrator, then the corresponding record in the computers table is updated.

Note:

Connect-It updates the record when the reconciliation proposal has been validated and when the value used for the update is the one that was validated in the reconciliation proposal. This value can be the value provided by Connect-It, the previous value in the database or another value that has been validated.

PEREGRINE

Index

U


Peregrine
SYSTEMS