HP Connect-It

ソフトウェアバージョン：3.81

AssetCenterデータベース統合ソリューション

法的制限事項

Copyrights

限定保証条項

機密コンピュータソフトウェア。

保有、使用、コピーを行うには、HPによる有効なライセンスが必要です。

FAR 12.211および12.212準拠。商用コンピュータソフトウェア、コンピュータソフトウェアマニュアル、技術データは、ベンダの標準商用ライセンスに基づき、米国政府にライセンス供与されています。

保証

HP製品およびサービスに対する保証は、当該製品およびサービスに付属の明示的保証規定に記載されているものに限られます。

本書のいかなる内容も当該保証に新たに保証を追加するものではありません。

HPは、本書中の技術的あるいは校正上の誤り、省略に対して責任を負いかねます。

ここに記載されている情報は、予告なしに変更されることがあります。

商標

- 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.

目次

はじめに

対象読者

本マニュアルは、AssetCenterへの統合の実行を必要とするユーザを対象としています

本マニュアルの使用法

「AssetCenterソフトウェア - 基本概念」の章

本章では、AssetCenterの一般的な概要を示し、属性、モデル、資産、およびポートフォリオ品目間に存在する関係について説明します。

「AssetCenterデータベースのマッピング」の章

本章では、AssetCenterデータベースマッピングに使用する主なアプローチを示し、次にコンピュータのマッピングに使用するさらに具体的なアプローチを示します。

「マッピング - 例」の章

本章では、Enterprise Desktop DiscoveryからAssetCenterデータベースへのマッピングについて詳しく考察します。

本書で使用される表記法

以下に本マニュアルで使用する表記法のリストを挙げます。

表記法	説明
JavaScriptコード	コードやコマンドの例
等幅文字	DOSコマンド、関数のパラメータ、またはデータフォーマット
...	省略されたコードまたはコマンド。
注意: **補足情報**	補足情報
重要項目: **以下の注意事項は...**	重要な情報
ヒント: **使用上のヒント...**	ヒント
警告: **警告**	非常に重要な情報
Object	Connect-Itインタフェースオブジェクト：メニュー、メニューエントリ、タブやボタンなど。

以下の表記法も併せて使用されます。

- 番号が振られたリストにある順番に従って実行する手順です。以下に一例を挙げます。
  1. 手順1
  2. 手順2
  3. 手順3
- すべての図と表には、その章と、章に現れる順番で番号付けられます。例えば、第2章の4番目の表は**表2-4**となります。

1 AssetCenterソフトウェア - 基本概念

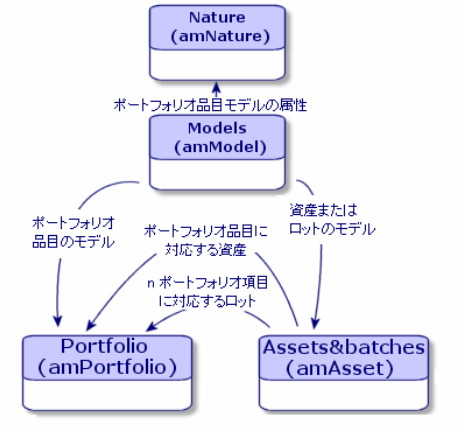
Connect-Itマッピングを実装するには、AssetCenterデータベース構造、ポートフォリオ品目の作成方法、およびポートフォリオ品目と従業員、契約やソフトウェアなどデータベース内のその他の要素との関係など、詳細にわたる知識が必要となります。

- AssetCenterマニュアル - 『コンセプトと導入』
- AssetCenterマニュアル - 『主要テーブル』
- AssetCenterマニュアル - 『はじめに』の「AssetCenterの概要」、「データベースの概要」
- AssetCenterマニュアル - 『ポートフォリオとソフトウェアライセンス』の「概要」

ポートフォリオ品目の相互依存性

ポートフォリオ、特にコンピュータの品目を作成する際、さまざまな作成に関するステップと条件に従うことが重要です。

ポートフォリオの構成はモデルを基にしています。各モデルは属性を基にしているため、モデルを作成する前に属性を作成する必要があります。



各属性が指定する内容を以下に挙げます。

- この属性にリンクされているモデルが、レコードを作成するテーブル。

  例：**コンピュータ**属性を使用して、ポートフォリオ品目テーブルにコンピュータを作成するのに使用するモデルを作成します。ポートフォリオ品目の作成に使用する属性の場合、第2の基準である管理条件の入力が必須です。
- ポートフォリオ品目モデルの作成に使用する各モデルに対し、動作のオプションを選択できます。

  例：**コンピュータ**属性では、動作オプション**接続可能**を有効にすると、接続ポートに関連するタブが表示されるようになります。

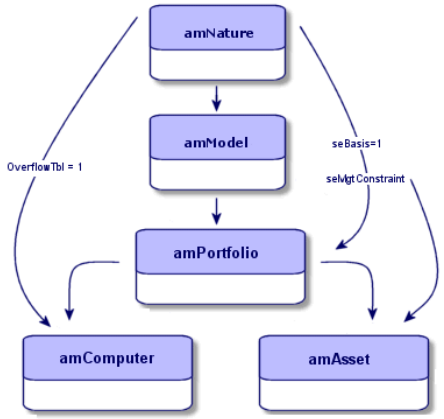
▶AssetCenterマニュアル - 『ポートフォリオとソフトウェアライセンス』の「概要」「属性：作成と動作オプション」

整合性の規則

以下の手順に従い、ポートフォリオ品目を作成します。

- 動作が**ポートフォリオ品目の作成**である属性を定義します。
- 動作がポートフォリオの作成である属性を基にモデルを定義します。

- 属性がポートフォリオ品目の作成であるモデルを基にポートフォリオ品目を定義します。



▶ AssetCenterマニュアル - 『物理データモデル』

管理条件

ポートフォリオ品目に定義する管理条件（属性テーブルで定義）を、Connect-Itマッピングでも考慮する必要があります。

管理条件を以下に挙げます。

- 固有資産タグ：固有資産タグを持つポートフォリオ品目は、個別にトラッキングされる品目です。資産タグは、サーバなど会計管理が必要な最も重要なポートフォリオ品目に使用します。
- 資産タグ：同一資産タグを共有するポートフォリオをロットにグループ化し、一括でトラッキングします。同一ロットに含まれる品目は、同一資産タグを共有します。このモードは、50台のモニタなど、個別にトラッキングする必要のない同一品目向きです。
- 個別管理しない：個別管理しない管理条件をポートフォリオ品目の属性として選択すると、資産タグをポートフォリオ品目に関連付けるかどうかを選択できます。資産タグのないポートフォリオ品目は正確なトラッキングを必要としないポートフォリオ品目向きです。これらの品目はトラッキングしないロットにまとめてグループ化され、資産テーブルには表示されません。消耗品のトラッキングは、その消耗品を消費する品目を経由することで間接的に行われます。

管理条件のタイプに応じて、作成先となるテーブルが異なります。

- 固有資産タグ、資産タグ、共有資産タグの場合、品目はポートフォリオ品目テーブルと資産テーブルに作成されます。

- 個別管理しないの場合、作成される品目はポートフォリオ品目テーブルにのみ記録されます。

▸AssetCenterマニュアル - 『ポートフォリオとソフトウェアライセンス』の「概要」

オーバーフローテーブル

特定のポートフォリオ品目は固有のフィールドを必要とします。これらのフィールドは、オーバーフローテーブルという特殊テーブルに格納されます。

1つのポートフォリオ品目レコードに、1つまたは複数のオーバーフローテーブルが指定されると、このレコードはポートフォリオ品目テーブルとオーバーフローテーブルの両方に同時に作成されます。例えば、資産テーブルとコンピュータテーブルの場合、いずれかのテーブルでレコードの追加や削除を行うと、もう一方のテーブルでも対応レコードの追加や削除が行われます。

オーバーフローテーブルによって、他のアプリケーションとAssetCenterとの統合が容易になります。例を挙げると、AssetCenterはネットワークインベントリ情報を統合し、この情報をコンピュータオーバーフローテーブルに入力できます。

主要なAssetCenterのオーバーフローテーブルを以下に挙げます。

- 資産テーブル
- コンピュータテーブル
- ソフトウェアのインストールテーブル
- モニタテーブル

このデータモデルにより、ポートフォリオ品目テーブルへのリンクと資産テーブルへのリンクがamComputerドキュメントに存在することになります。

▸AssetCenterマニュアル - 『ポートフォリオとソフトウェアライセンス』の「概要」、「オーバーフローテーブル」

2 AssetCenterデータベースのマッピング

AssetCenterデータベースのConnect-Itマッピングでは、アプリケーションに備わっているその他のデモンストレーションシナリオに類似した、同一の全体構造を使用します。

Asset Managementコネクタのカスタマイズ

Asset Managementコネクタは、AssetCenterデータベースと併用するように特別に開発されています。

コネクタを使用してシナリオファイルを開くと、構成ファイルが以下の順序で読み込まれます。

1 Connect-Itインストールフォルダ**config\shared**のデフォルト構成

2 Connect-Itインストールフォルダ**config\ac**のデフォルト構成

3 Connect-Itインストールフォルダ**scenario\[シナリオ名]**のユーザ構成

注意:

関数名は一意である必要があります。複数の関数が同じ名前の場合、エラーメッセージが表示されます。

▶ Connect-Itマニュアル - 『コネクタガイド』の「HP コネクタ」、「Asset Managementコネクタ」

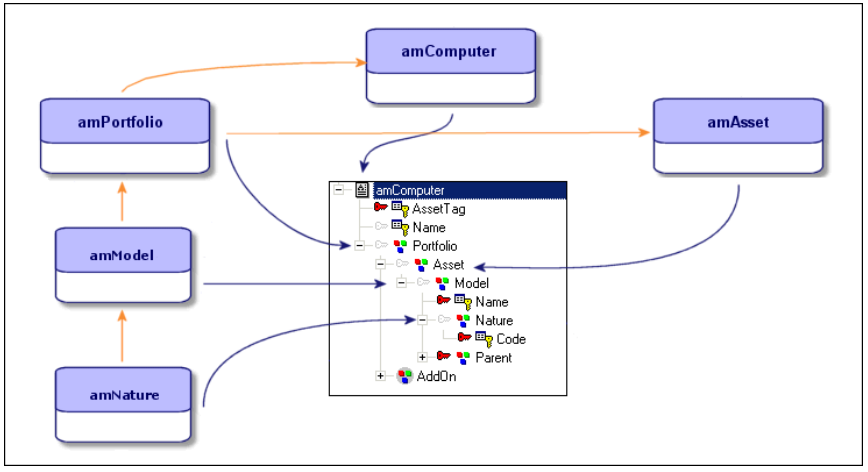
使用言語

言語依存性を解消するため、Connect-Itスクリプトによって呼び出され、使用されて、そのままターゲットデータベースに挿入される共通文字列を保存します。これらの文字列は、.strファイルに格納され、**PifStrVal**関数が呼び出します。

主なアプローチ

Asset Managementコネクタが生成、取り込みを行うドキュメントタイプは、AssetCenterデータベーススキーマの構造の解釈です。

このため、マッピングは品目の作成条件を反映する必要があります。



マッピング - 概要

ソースドキュメントの要素とターゲットドキュメントの要素が互いにリンクされると、マッピングが作成されます。

シナリオの処理

シナリオを実行すると、以下の操作が実行されます。

1 ソースコネクタが生成するドキュメントタイプが作成されます

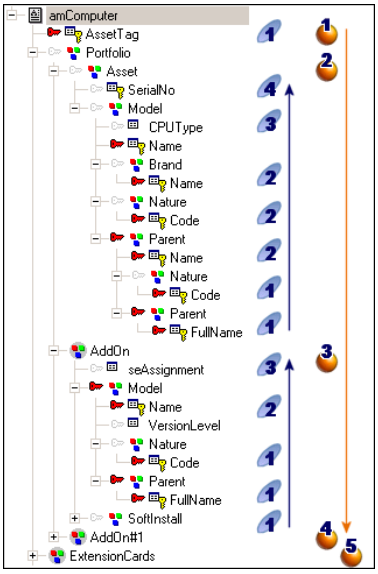
2 マッピングからドキュメントタイプが作成されます。

3 ターゲットコネクタの照合更新が実行されます。マッピングによるドキュメントの要素と、データベース中のレコードに存在するフィールドとが比較されます。

4 照合更新スクリプトがターゲットコネクタに適用されます。

▶ Connect-It - 『ユーザガイド』の「統合シナリオのインプリメンテーション」、「ドキュメントタイプのマッピングの定義」

マッピングの処理



マッピングを構成する複合要素の処理順序は以下のとおりです。

1 ドキュメントタイプを構成する第1レベルの要素：フィールド（1）

2 ドキュメントタイプを構成する第2レベルの要素：構造体（2）
構造体はコレクションの前に処理されます。コレクションが構造内に存在する場合、ドキュメントタイプ全体の全構造の処理後に処理されます（1、2、3、4、5）。

3 ドキュメントタイプを構成する第3レベルの要素：コレクション（3）

重要項目:

Use the parent ID as a reconciliation keyオプションを使用する場合、この処理順序を実装する必要があります。このオプションは、2つのポートフォリオ品目を接続するリンクをたどるのに使用され、親アイテムの照合更新キーを使用します。このオプションは、**Follow the link**オプションと逆の動作をします。

▶ Connect-It -『コネクタガイド』の「コネクタのルール（ディレクティブ）」

AssetCenterの管理条件

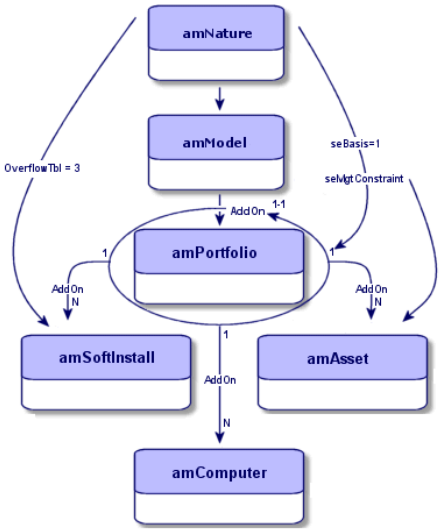
品目に設定された管理条件に応じて、インベントリマッピングとマイグレートするデータがポイントするテーブルは異なります。

マイグレートする情報の対象に応じて、マッピングが使用するドキュメントタイプは異なります。

- コンピュータ関連の情報の場合、amComputerドキュメントタイプ
- ソフトウェアの場合、amSoftInstallドキュメントタイプ

> **注意:**
>
> **コンピュータに関連付けられたソフトウェアのインストールは、特殊な処理方法を使用し、amComputerドキュメントタイプで表示されるAddOnコレクションによって表現されます。**

同一ドキュメントタイプ内で、異なるAssetCenterテーブルを結合するリンクを使用する全情報の管理が可能です。例えば、部署と従業員に関係する情報を、ポートフォリオ品目テーブルにあるリンクを経由して移動できます。



これらのドキュメントタイプそれぞれに対し、適用される管理条件により、複合ポートフォリオ品目へのリンクが作成されます（ポートフォリオ品目をデフォルトで作成する属性、および追加情報がリンクされたオーバーフローテーブルに入力されています）。

- Connect-It -『ユーザガイド』「統合シナリオのインプリメンテーション」の「ドキュメントタイプのマッピングの定義」
- AssetCenterマニュアル -『管理』の「データベースの標準記述ファイル」

AssetCenterデータベースへのマッピングの準備

マッピングのキー要素を決定する前に実行が必要な操作を以下に挙げます。

- AssetCenterコネクタが使用するドキュメントタイプを表示し、AssetCenterで使用されない複合要素（フィールド、構造体、コレクション）などの情報を削除します。
- マッピング用に生成または取り込みが行われたドキュメントタイプの重要な情報（構造体、コレクション）を識別します（AssetCenterで使用されるテーブルのフィールドとリンクを識別します）。

マッピングのキー要素の決定

AssetCenterデータベースモデルで使用される整合性の規則があるため、Asset Managementコネクタを使用するマッピングのキー要素は、マッピングが行う必要のある操作に応じて異なります。このため、入力する必須フィールドも異なる場合があります。

クライアントデータベースのカスタマイズと同様、関連する専門分野も考慮する必要があります。例を挙げると、専門分野の規則により、ユーザに割り当てられる要素が在庫に入ることが必要となる場合、この品目がユーザに直接割り当てられることがないよう、マッピングはこの規則を考慮する必要があります。

AssetCenterで品目を作成する際の条件を以下に挙げます。

- AssetCenterに固有な整合性の規則の遵守
- 照合更新キーを構成する項目の識別
- 取得された情報に基づく、挿入する項目の識別

先に説明した通り、ポートフォリオ品目の作成の前提条件に従い、以下のマッピングでAssetCenterでレコードの作成が必要となる主な項目を説明します。

▶ Connect-It - 『ユーザガイド』の「統合シナリオのインプリメンテーション」、「生成用または取り込み用ドキュメントタイプの定義」

属性の作成

AssetCenterで、作成する属性は以下の情報を必要とします。

- amNatureルートドキュメント：Name、Code要素

  属性の名前ではなく属性のコードを使用することを推奨します。属性の名前はインストールされた言語バージョンによって変化する場合があるためです。

マッピングで値が定義されていない場合、これらのフィールドはAssetCenterのスクリプトを経由して自動的に入力されます。

重要項目:

属性の作成にはConnect-Itを使用することは推奨しません。AssetCenterで行ってください。

しかし、属性をマッピングで作成する必要がある場合、マッピングで属性の作成に必要となる全フィールドを使用する必要があります。

デフォルトで、管理条件として固有資産タグを持つポートフォリオ品目が属性により作成されます。

▸ AssetCenterマニュアル - 『ポートフォリオとソフトウェアライセンス』の「ポートフォリオ品目」、「属性」

モデルの作成

AssetCenterでは、作成するモデルは属性を必要とします。

amModelドキュメントタイプの場合、モデル作成時に以下の項目が必要です。

- amModelルートドキュメント：Name、Barcode要素

  重要項目:

  モデルの名前は一意ではありません。FullNameのみが一意です。

- Nature構造：Name、Code要素

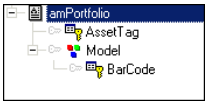
上記要素のみが必須です。マッピングに他の値が指定されていない場合、**Barcode**フィールドが自動的に値を取得します。

注意:

しかし、追加のオプションとなるフィールドを使用して、AssetCenterデータベースに関連情報を入力することができます。

▸ AssetCenterマニュアル - 『ポートフォリオとソフトウェアライセンス』の「ポートフォリオ品目」、「モデル」

ポートフォリオ品目の作成



AssetCenterで、作成するポートフォリオ品目は以下の情報を必要とします。

- amPortfolioルートドキュメント：
- Model構造：Barcodes

上記要素のみが必須です。マッピングに他の値が指定されていない場合、**Barcode**フィールドが自動的に値を取得します。

注意:

Follow the linkオプションを使用すると、第1レベルの要素ではなく、第2レベルの要素（資産のシリアル番号など）を照合更新キーとして選択できます。

▶ Connect-Itマニュアル - 『HP コネクタガイド』の「Asset Management コネクタ」、「Asset Managementコネクタの生成用ルール」

照合更新キーの決定

照合更新キーをどのように決定するかは、会社で決められている資産管理方法によって異なります。例えば、各資産が調達サイクル内で個別に監視されている場合、明らかにコンピュータの識別子やそのシリアル番号を照合更新キーとして選択することになります。

しかし、調達サイクルがAssetCenterで監視されていない場合、時間が経過しても変化しない要素を照合更新キーとして定義する必要があります。これは、コンピュータのネットワーク名（**amComputer**ドキュメントタイプの**Name**）やコンピュータのMACアドレスです。

照合更新キーを決定する際、以下の項目を決定する必要があります。

- ソースデータベースと、AssetCenterデータベース内の双方に存在する、1つまたは複数のフィールド
- 品目のライフサイクルを通して変化しない値を持つフィールドである、1つまたは複数個の「ソリッド」な照合更新キー

ヒント:

照合更新の対象となる要素を明確にするため、要素のFullNameを使用して照合更新を行うことが頻繁にあります。要素のFullNameを.strファイルに保存し、PifStrVal関数を使用してそのFullNameを呼び出すことを推奨します。こうすることで、要素のFullName変更時にマッピング全体を変更する必要となる場合が減少します。

▸ Connect-It - 『ユーザガイド』の「統合シナリオのインプリメンテーション」、「ドキュメントタイプのマッピングの定義」、「照合更新キー」

照合更新キーの動作

照合更新キーが定義されている場合、生成されるドキュメントを一意に識別するのにキーを含むフィールドを使用し、ターゲットテーブル内のフィールドと比較します。

リンク上の照合更新キー

照合更新キーがリンク（構造体）に設定されている場合、これまでの値は置換されず、新しい値が挿入されます。

代替照合更新キーの決定

最初のキーでエラーが発生した場合、Connect-Itでは、1つまたは複数の照合更新キーを定義できます。

照合更新キーの決定には、以下のような2つの目的があります。

- 特定期間内に、時間変化して主キーの伝播を可能にする有効な値の検索と定義（例えば、特定のコンピュータに一意な識別子が存在しない場合、コンピュータ名やMACアドレスを使用して代替照合更新ソリューションを実装できます）。
- ソースにある固定キーを含む数フィールド用の、ポートフォリオ品目のタイプによって異なる特定キーの検索と定義。これにより、ポートフォリオ品目のタイプに応じて個別のマッピング（ルータ、コンピュータ）に処理させます。

▸ Enterprise DiscoveryからAssetCenterへのシナリオ [献 29].

▸ Connect-It - 『ユーザガイド』の「統合シナリオのインプリメンテーション」、「シナリオオプションの編集」、「Display」

照合更新キーに関連付けれたマッピングスクリプト

マッピングスクリプトを照合更新キーに関連付け、値に応じた動作を定義できます。

マッピングスクリプトを作成して、照合更新キーとして使用する要素に値を仮割り当てします。

以下の2つの場合が考えられます。

- 挿入される値が一意である。
- 挿入される値がモデルまたは属性で既存である。

照合更新キーを使用する際の条件を以下に挙げます。

- 照合更新キーとして使用する各フィールドで、選択した要素がAssetCenter内に値を持つ。

  利用可能な値が存在しない場合、照合更新は実行されず、エラーがドキュメントログに保存されます。

  注意:

  NULL値（空の文字列）を使用できます。

- NULL値を挿入すると、整合性の規則に違反する場合がある（例えば、資産の資産タグは常に値を持つ必要るあり）。

ヒント:

Show queries in tracking linesオプションを使用すると、AssetCenterデータベースに送信されるクエリが見やすくなります。

▸ **Connect-It - 『ユーザガイド』、「統合シナリオのインプリメンテーション」、「モニタの定義」**

▸ Connect-It - 『ユーザガイド』の「統合シナリオのインプリメンテーション」「シナリオオプションの編集」「コネクタ」

ソース値が存在しない場合

ソースデータベースに値が存在しないときに、ターゲットデータベースに一意の値を挿入するには、以下の手順を実行します。

- マッピングスクリプトを作成する
- グローバル関数を使用する

グローバル関数では、マッピングの別の場所で同一スクリプトを共有し、読みやすさを改善できるため、グローバル関数を使用することを推奨します。

例えば、グローバル関数は、コンピュータのネットワーク識別情報（名前、コンピュータが所属するグループとドメイン）を連結して識別子を作成できます。

▸ Connect-It - 『ユーザガイド』の「統合シナリオのインプリメンテーション」、「マッピングスクリプトの定義」、「関連ファイルの編集」

照合更新キーの一意な値の定義

マッピングスクリプトでデフォルト値を定義できます。

例えば、インベントリツールはコンピュータに関連する情報を送信しますが、AssetCenterデータベースにこのコンピュータに関連するモデルがないとします。この場合、割り当てられるべきデフォルト値を割り当て、値が設定されていない場合はUnknownモデルを使用するよう、スクリプトが定義します。

▶ Connect-It -『ユーザガイド』の「統合シナリオのインプリメンテーション」、「マッピングスクリプトの定義」

照合更新スクリプトの実行

マッピングスクリプトの後に照合更新スクリプトを適用します。照合更新スクリプトを使用してAsset Managementコネクタが取り込むドキュメントタイプに以下のアクションを実行します。

- 更新（ターゲットデータベースに値が存在する場合）
- 挿入（ターゲットデータベースに値が存在しない場合）

照合更新スクリプトを使用すると、以下の結果が得られます。

1 レコードの、更新対象フィールドのリスト項目が削除されます。

2
- 更新モードでは、既存の値が保持されます。
- 作成モードでは、デフォルト値が使用されます。

NULL値を持つことのないAssetTagなどの照合更新キーを持つ項目に対して照合更新を実行する場合のみ、この方法で照合更新スクリプトを使用できます。

▶ Connect-It -『コネクタガイド』の「コネクタのルール（ディレクティブ）」「照合更新」

照合更新キャッシュ

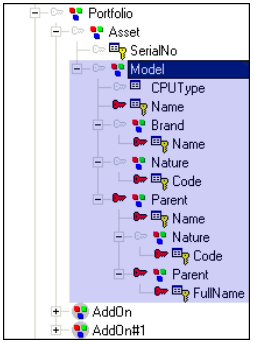
データベースクエリの回数と実行時間を減少させるのに照合更新キャッシュが使用されます。キャッシュはメモリに格納されています。

照合更新キャッシュが有用となる場合を以下に挙げます。

- マッピングで定数が使用されている場合
- マッピングが、モデル（amModel）や属性（amNature）テーブルといった比較的小規模のテーブルで実行される場合
- マッピングが、更新のないテーブルで実行される場合

▶ Connect-Itマニュアル -『コネクタガイド』の「HP コネクタ」「Asset Management コネクタ」

以下のスクリーンショットは、インベントリシナリオ中に照合更新キャッシュが関連している構造のセットを示しています。



照合更新キャッシュにより、クエリ数を減少させることで、特定セッションのシナリオパフォーマンスが最適化されます。

照合更新キャッシュは以下の要領で機能します。

1 Connect-Itが開きます
2 照合更新キーの値がメモリに格納されます。
3 照合更新キーに対応する一意な識別子がキャッシュに格納されます。
4 マッピングで使用されている各キャッシュされた要素については、クエリは送信されません。

> **注意:**
>
> **項目やドキュメントがキャッシュに格納されると、値が取得され、現在のセッションでは最新ではなくなります。**

5 定義されているドキュメントの最大数に達すると、キャッシュを削除します。
6 キャッシュはセッション終了時に削除されます

キャッシュに格納されるドキュメントの最大数は、コネクタのオプションで定義します（Edit/Options/Connector）。

並列化

並列化は、ドキュメントタイプを取り込む全コネクタに適用されます。並列化では、1コネクタを複数プロセスに複製し、ドキュメント取り込みを並列に処理します。

並列化に伴うパフォーマンスの改善は、ドキュメント生成の速度とドキュメント取り込みの速度によって異なります。例えば、シナリオによるドキュメント生成がドキュメント取り込みより遅い場合、並列化された取り込みによるメリットは

ありません。データベースアーキテクチャやネットワーク問題といったその他の要因もパフォーマンスに影響を与えます。

デッドロック

2つのプロセスが同一レコードにアクセスを試みる場合、データベースへの書き込みのデッドロックが発生する場合があります。あるプロセスがレコードにアクセスし、他のレコードのアクセスを妨げているような場合です。

以下のような方法を実行して、デッドロックを防ぐことができます。

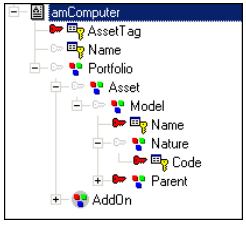
- Connect-Itで、1トランザクション当たりのドキュメント数を制限する（100未満）。
- Connect-Itで、モデルテーブルなどの同一テーブルへの同時アクセスを避ける。
- Connect-Itで、シナリオを2つのマッピングに切り分ける。たとえば、最初のシナリオでは、リポジトリを作成（モデル、従業員、場所や属性などの挿入）し、並列化モードでは動作しない。2番目のマッピングでそのリポジトリからデータを読み込み、新規データを並列化モードで挿入する。
- Connect-Itで、書き込みアクセスを避けるため、キャッシュを使用する。
- AssetCenterでカウンタの使用を避ける。特にAssetCenterウィザードでカウンタをオーバーフローテーブルに移動しない。
- 使用しているDBMS（DB2、Oracleなど）の仕様に合わせた操作を行う。

▶ AssetCenterマニュアル - 『Tuning』、「Tuning the database」の「Eliminating locks and deadlocks」

特別なアプローチ - コンピュータのマッピング

コンピュータの作成

コンピュータを作成する際のマッピング構造を以下に挙げます。



このマッピングタイプでは、コンピュータは資産と自身がモデルに依存するポートフォリオ品目にリンクされています。

ポートフォリオ品目作成時に必要となる要素を以下に挙げます。

1 ルートドキュメントamComputer：AssetTag要素、Name
2 Portfolio構造
3 Asset構造
4 Model構造

- Portfolio構造

  ポートフォリオ品目をオーバーフローテーブルのレコードにリンクできます。この動作は、ポートフォリオ品目を作成するモデルの属性で定義されます。このため、ポートフォリオ品目テーブルは、コンピュータテーブルおよび資産テーブルにリンクされます（管理条件によってリンクされるテーブルは異なります）。

  重要項目:

  資産が基本とするモデルの属性はコンピュータである必要があります。

- Asset構造

  この構造が必要となるのは、ポートフォリオ品目の作成や更新の際に、資産のシリアル番号やモデルへのリンクなどの要素が必須の場合のみです。

  属性がポートフォリオ品目の作成を必要とする場合、リンクされたポートフォリオ品目の作成を定義することもできます（コンピュータ、モニタ、ソフトウェアのインストール、電話）。固有資産タグや共有資産タグタイプの管理条件は、ポートフォリオ品目の作成と同時に行われる資産テーブルにリンクされたレコードの作成に関連付けられています。作成後、資産の属性は変更できません。

  資産テーブルには、ポートフォリオ品目のAssetTagも含まれます。データモデルで使用されている場合、ポートフォリオ品目のマッピング時にこのテーブルを使用する必要があります。

- Model構造

  資産テーブルでのレコードの作成は、ポートフォリオ品目の場合と同様、モデルに基づいて行われます。このため、レコードの作成に使用されるモデルもマッピングに含まれます。

コンピュータ作成時の必須キーの選択

AssetCenterにある「コンピュータ」タイプのレコードの作成や更新の際に必要となる要素をマッピングする際、照合更新キーは以下の要素に対応します。

- **AssetTag**要素：

コンピュータの識別子（**Asset Tag**フィールド）は定数です。この値は長期間にわたって一定であり、コンピュータを一意に識別できます。この項目に照合更新キーを定義することで、品目の基準を一定値にできます。

- **Name**要素：ポートフォリオ管理モデルがコンピュータやバーコードのような固有の識別子に基づいていない場合、照合更新キーとして機能するほかの要素を選択します。コンピュータ名（**Name**フィールド）は、ITポートフォリオを固有に識別する代替メソッドです。このフィールドはマッピングスクリプトで値が指定されていないか、マッピングに含まれていない場合、AssetCenterエージェントによって自動的に入力されます。

> **重要項目:**
>
> **ネットワーク上で識別されているコンピュータ名は変更できません。変更してしまうと、インベントリツールがネットワークの新規インベントリを実行する際、新しいコンピュータが作成されてしまいます。**

- **PhysicalAddress**および**TcpIpAddress**要素：識別子や一意な名前がない場合、これらの要素を照合更新キーとして使用できます。
- **Name**（Model.Name）要素：

モデルの名前は必須です。ポートフォリオ品目は、それ自体がモデルにリンクされている資産にリンクされているため、モデルの**Name**フィールドで照合更新が実行されます。

要約すると：

- コンピュータを一意に識別するには、AssetTagが使用されます。
- コンピュータを作成するには、資産タグ、モデル名、およびポートフォリオ品目テーブルとモデルテーブル間のリンクが必要となります。

マッピングスクリプト

各照合更新キーのマッピングスクリプトを定義します。

このマッピングスクリプトは以下のように定義します。

- 2つの要素がリンクされている場合、Connect-Itを経由して直接的に定義する
- BASICスクリプトを経由して手動で定義する

▸ Connect-It -『ユーザガイド』の「統合シナリオのインプリメンテーション」、「マッピングスクリプト」

Follow the link

任意の選択した構造からアクセス可能なこのオプションは、以下の状況で有用です。

- オーバーフローテーブルに対して使用する場合。参照テーブルにある項目のリンクをたどることで、リンクされたテーブルから情報を取得できます。
- 信頼性のある照合更新キーを定義できない場合（たとえば、照合更新にEmployee IDを使用できず、NameとFirst Nameのみを使用する場合など）。

▶ Connect-Itマニュアル - 『コネクタガイド』の「HP コネクタ」、「Asset Management コネクタ」、「Asset Managementコネクタの生成用ルール」

コンピュータに関連する情報の作成や更新を行うマッピングの場合、Follow the linkオプションは以下の構造で使用できます。

- Portfolio

  amComputerテーブルはamPortfolioテーブルのオーバーフローテーブルです。この場合、amPortfolioはamComputerテーブルに一対一にリンクされているため、amPortfolioへのリンクの整合性を確認するクエリを実行する必要はありません。

  コンピュータが存在する場合、ポートフォリオ品目が存在するため、要求は行われません。

  注意:

  AssetCenter固有の整合性の規則により、コンピュータ作成時にエージェントがトリガされ、ポートフォリオ品目テーブルにある対応レコードが自動的に作成されます。

- Asset

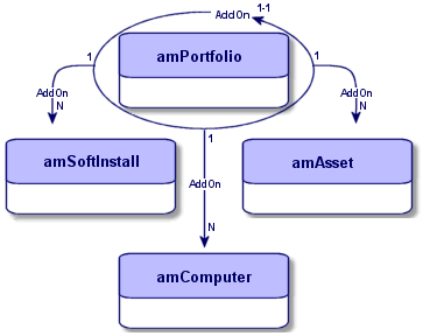
  検索されるプロパティはモデルテーブルに対応します。1モデルのみがポートフォリオ品目に対応します。

  検索されるプロパティは属性テーブルに対応します。1属性のみがモデルに対応します。

コンピュータにリンクされた情報

タイプが「コンピュータ」であるレコードがAssetCenterデータベースに作成されると、そのコンピュータにリンクされた情報が、隣接テーブルに記録されます。これは、ネットワークカード、拡張カード、または物理的なハードディスクに関連する情報の場合です。AssetCenterテーブルにある名前に対応する名前を持つコレクションや、AddOnタイプコレクションにあるamComputerドキュメントタイプマッピングについて、この情報を表示できます。

AddOnコレクション



Component（AddOn）リンクに対応するAddOnタイプコレクション、ポートフォリオ品目テーブル（amPortfolio）で表示できるAddOnタイプコレクション、OwnCopyリンクタイプは、以下の操作実行時に使用します。

- ツリー構造の定義。
- ポートフォリオ品目の、オーバーフローテーブルの一部である別のポートフォリオ品目へのリンク。

AddOnコレクションは、以下の情報をグループ化します。

- ソフトウェアのインストールに関連する情報：AddOnコレクション、SoftInstall構造
- モニタに関連する情報：AddOn#1コレクション、Model構造

注意:

AddOnコレクション（AddOn#1、AddOn#2など）の名前は、マッピングの記述順序によって異なります。

これらのコレクションは、ポートフォリオ品目テーブルのレコードの**Component**タブで定義されているリンクに対応します。

AddOnタイプコレクションの使用方法は、ユーザが定義した管理タイプによって異なります。

Use the parent ID as the reconciliation keyオプション

このオプションは、親ポートフォリオ品目に属するAddOnタイプ要素のみを処理するのに、照合更新で使用します。

ExtensionCards、LogicalDrives、NetworkCards、PhysicalDrivesの各コレクション

コンピュータにリンクされ、AddOnコレクションに定義されたカテゴリには属さない情報は、以下のテーブルに記録されます。

- amExtensionCards
- amLogicalDrives
- amNetworkCards
- amPhysicalDrives

これらのテーブルには、コンピュータに関する補足情報が含まれますが、ポートフォリオ品目テーブル（amPortfolio）のオーバーフローテーブルではありません。

これらのテーブルは、コンピュータテーブルで表示できる、**Extension**、**Disks**、および**Network**タブに対応します。

3 マッピング - 例

本章では、用例シナリオである、Enterprise Discoveryコネクタ提供によるedac.scnのマッピング構造について、詳しく説明します。

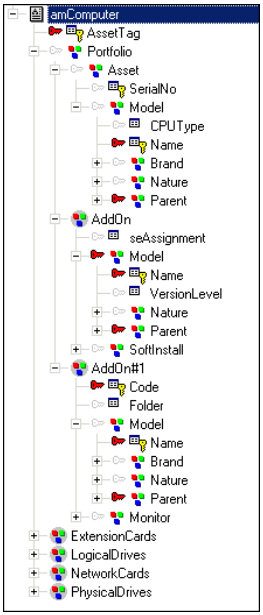
Enterprise DiscoveryからAssetCenterへのシナリオ

本シナリオで、Enterprise Network Discoveryデータベースから AssetCenterデータベースにデータを転送できるようになります。

以下で説明するマッピングはNetwork-Devices *Structure-amPortfolioDst*マッピングです。

マッピング構造

マッピングは以下のように構造化されています。



この構造は、前の章で説明したスキーマに従います。

マッピングは作成された品目であるコンピュータから開始し、次に作成に使用される要素（ポートフォリオ品目、モデル、属性）などへのリンクが行われます。これは、品目の作成と挿入に必要となる全要素（属性、モデル）を呼び出す、AssetCenterで使用される作成処理とは反対です。

▶ マッピングの処理 [献 14].

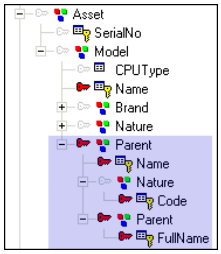
edac.scnマッピングには、**amComputer**ドキュメントタイプのその依存関係が関係します。

- ポートフォリオ品目
- モデル
- 属性
- ブランド
- ソフトウェアのインストール
- モニタ
- 拡張カード
- 内部デバイス

マッピングにある各複合要素に対し、AssetCenterの1テーブルが存在します。

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

Parent構造



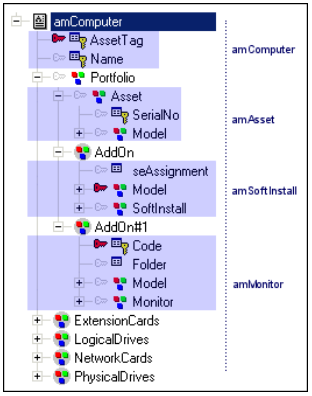
AssetCenterにあるいくつかのモデルは、同一名を共有しながら異なる属性を持つことができます。名前のみを使用してモデルを区別することはできないため、モデルの親やモデルに関連付けられた属性を識別することで、モデルを区別できます。例えば、EAIモデルからはその属性に応じて、ソフトウェア、またはソフトウェアライセンスとインストールが作成されます。

AssetCenterでは、sysComputerやMODEL_WORKSTATION_AC44といった定数値が使用されます。これらの値がマッピングスクリプトで使用されます。インベントリにより、AssetCenterデータベースには存在しない値が返されると、これらの値により、コンピュータの作成に使用された属性にリンクされたモデルにコンピュータが割り当てられ、コンピュータが挿入されます。

照合更新要素の識別

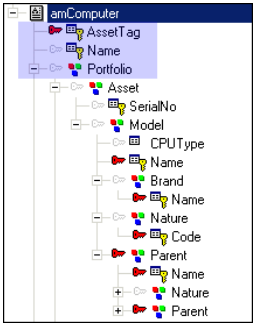
マッピングでは、コンピュータの作成に必要となる照合更新要素は、コンピュータにリンクされた情報の作成に必要となる要素と区別する必要があります。通

常、これらの要素（構造体、コレクション）は、AssetCenterに表示されるテーブル名に対応します。



コンピュータ、ポートフォリオ品目、および資産間に相互依存性が存在するため、相互依存性を作成する要素がコンピュータの作成や更新マッピングに存在します。

マッピングでの要素の役割を区別することが重要です。要素を使用して、照合更新キーを要素に割り当てても、マッピング内でこの照合更新キーが別のキーと同じ役割を果たさないためです。



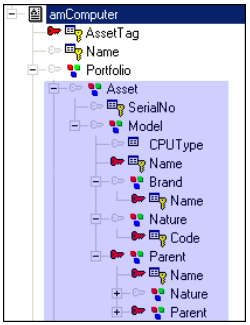
このため、複合要素**AssetTag**に配置されるキーを使用して、識別子に応じてコンピュータを一意に取得しますが、コンピュータの作成は行いません。コンピュータの作成や更新の方法は、コンピュータの依存性によって異なります。同じ方法で、複合要素**Brand.Name**に設定されている照合更新キーは、この要素の作成や更新を行いますが、コンピュータ情報自体の整合性は確認しません。

照合更新キーの選択

このシナリオで解決する問題を以下に挙げます。

- 最初のネットワークインベントリが実装され、コンピュータに識別子が存在しない場合に、コンピュータの識別子の作成方法を決定します。
- 長い期間にわたって変化せず、スキャンされた品目の値とAssetCenterデータベースに保存された値との照合更新を可能とする、信頼性のある照合更新キーを選択します。

このシナリオの照合更新要素は、このマニュアルで説明されている以下の一般構造に従います。



Enterprise DiscoveryからAssetCenterへのシナリオでの、照合更新キーの選択肢を以下に挙げます。

- **AssetTag**要素：インベントリシナリオで、ITポートフォリオの項目にリンクされた識別子により、この項目を一意に識別できます。

  AssetTag要素を主照合更新キーとして定義すると、ネットワークインベントリ情報の取得時に、各レコードはそのAssetTagで識別されます。
- **Name**（Model.Name）要素：単一モデルのみにリンクできるポートフォリオ品目、コンピュータです。モデルの名前を照合更新キーとして選択します。
- **Name**（Brand.Name）要素：インベントリツールは、ブランドにリンクされている情報を取得します。各インベントリ後に新ブランドを追加してしまうことを防ぎ、値とAssetCenterデータベースにある既存の値との照合更新を行うため、ブランド名を使用して照合更新を行います。
- **Code**（Nature.Code）要素：属性により、モデルが作成する対象が定義されます（ポートフォリオ品目、資産など）。属性のコードは、属性を一意に識別するため、照合更新キーとして使用されます。
- **Name**（Parent.Name）要素：モデルを他のモデルから構成し、階層を構成できます。親モデルの名前を照合更新キーとして使用します。
- **Code**（Nature.Code）要素：属性により、モデルが作成する対象が定義されます（ポートフォリオ品目、資産など）。属性のコードは、属性を一意に識別するため、照合更新キーとして使用されます。

- **FullName**（Parent.FullName）要素：FullNameは、階層でのモデルの所属先を定義するのに最後に使用した項目です。FullNameは一意であるため、照合更新キーとして使用します。

代替照合更新キー

最初のキーでエラーが発生した場合、キーセットに代替照合更新キーを定義することができます。このシナリオでは、3つのキーセットである、主キーのセット1つ、代替照合更新キーのセット2つを定義できます。

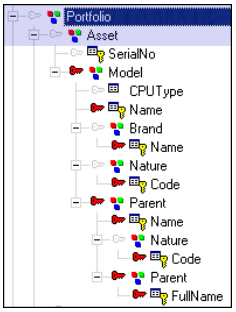
定義されるキーを以下に挙げます。

- **PhysicalAddress**：最初の照合更新キー（AssetTag）でエラーが発生した場合、MACアドレスを経由してネットワークでコンピュータを識別できる場合、この要素を2番目の照合更新キーとして定義します。
- **TcpIpAddress**：2番目の照合更新キー（PhysicalAddress）でエラーが発生した場合、IPアドレスがネットワークで変化しない要素である場合、この要素を3番目の照合更新キーとして定義します。

▶ 代替照合更新キーの決定 [献 19]

Follow the linkオプションの選択

項目同士が一対一リンクでリンクされている場合、クエリを送信してテーブルに関する整合性を確認する必要はありません。**Follow the link**オプションを有効にして、クエリを実行しないようにしてください。



Follow the linkオプションを有効にする構造を以下に挙げます。

- Portfolio
- Asset

このため、これらの構造のサブ項目には、照合更新キーは定義されません。

照合更新キーに関連付けれたマッピングスクリプト

照合更新キーに関連付けられているスクリプトを使用すると、同一データフォーマット処理を踏襲して、AssetCenterデータベースでの値の挿入や更新を実行できます。

- **AssetTag**要素のスクリプト：

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

このスクリプトは、以下のフィールドの値をパラメータとして使用して、**EDDIGetComputerModel**関数を呼び出します。

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

最初の3つのパラメータに値が存在しない場合、関数は最後のパラメータの値を返します。

次に、値は**ToSmart**命令でフォーマットされます。

- **Name**（Model.Name）要素のスクリプト：

```
UCase(EDDIGetACAssetTag ([hwAssetData.hwAssetTag], [hwNetworkData.hwNetworkNames.hwWorkgroupName], [hwNetworkData.hwNetworkNames.hwLocalMachineID], [NMID.Appliance.Appliance_ServerID], [NMID.NMID_NMID]))
```

このスクリプトは、以下のフィールドの値をパラメータとして使用して、**EDDIGetAssetTag**関数を呼び出します。

- hwAssetData.hwAssetTag
- hwNetworkData.hwNetworkNames.hwWorkgroupName
- hwNetworkData.hwNetworkNames.hwLocalMachineID
- NMID.Appliance.Appliance_ServerID
- NMID.NMID_NMID

次に、値は**UCase**命令でフォーマットされます。

- **Name**（Brand.Name）要素のスクリプト：

```
Dim strBrand As String

strBrand = EDDIGetComputerManufacturer ([hwSMBIOS.hwSmbiosSystemInformation(0).hwsmbiosSystemManufacturer], [CompanyHW.Company_Name])
If strBrand = "" Then
```

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

このスクリプトは、以下のフィールドの値をパラメータとして使用して、**EDDIGetComputerManufacturer**関数を呼び出します。

- hwSMBIOS.hwSmbiosSystemInformation(0).hwsmbiosSystemManufacturer]
- CompanyHW.Company_Name

名前は、**EDDIGetComputerManufacturer**関数が実行する連結となります。この関数が値を返さない場合は、UNKNOWNモデルが作成されます。値が**EDDIGetComputerManufacturer**から取得された場合は、その値は**ToSmart**関数でフォーマットされます。

- **Code**（Nature.Code）要素のスクリプト：

```
"sysComputer"
```

この値は、コンピュータテーブル経由でのコンピュータに関連付けられた資産の自動作成中に使用される属性のコードを定義します。

- **Name**（Parent.Name）要素のスクリプト：

```
Dim strBrand As String

strBrand = EDDIGetComputerManufacturer ([hwSMBIOS.hwSmbiosSyst
emInformation(0).hwsmbiosSystemManufacturer], [CompanyHW.Compan
y_Name])
If strBrand = "" Then
strBrand = PifStrVal("UNKNOWN")
End If
```

親項目の名前は、**EDDIGetComputerManufacturer**関数が実行する連結となります。この関数が値を返さない場合は、UNKNOWNモデルが作成されます。値が**EDDIGetComputerManufacturer**から取得された場合は、その値は**ToSmart**関数でフォーマットされます。

- **FullName**（Parent.FullName）要素のスクリプト：

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

識別子MODEL_WORKSTATION_AC44を使用して、すべてのサブモデルのデフォルト値が定義されます。この識別子により、ローカル言語に対応する文字列が使用可能になります。このため、あるインベントリされた品目の各サブアイテムが親子リンクによってモデルにリンクされ、MODEL_WORKSTATION_AC44識別子の値を取ります。

▶ Connect-It -『ユーザガイド』の「統合シナリオのインプリメンテーション」、「マッピングスクリプトの定義」、「関連ファイルの編集」

▶ Connect-Itマニュアル -『Programmer's reference』

マッピングに関連するグローバル関数

インベントリ中にこの情報が存在しない場合、このマッピングで、必須AssetCenterフィールドの情報を送信するのに主に使用されるいくつかの関数が開発されています。

例えば、**EDDIGetNMID**関数はコンピュータの一意な識別子を作成します。

以下の関数の説明には、識別子の仮作成に使用する関数が主に記載されています（一部網羅されていない関数があります）。

- **EDDIGetNMID**関数：

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

この関数には、以下の2つのパラメータが必要です。

- strServerID
- strNMID

パラメータが連結され、一意の識別子が返されます。

- **EDDIGetACAssetTag**関数：

```
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
```

この関数ではいくつかのパラメータが期待されます。最初の4つのパラメータが値を返さない場合、最後のパラメータであるstrNMIDの値が使用されます。

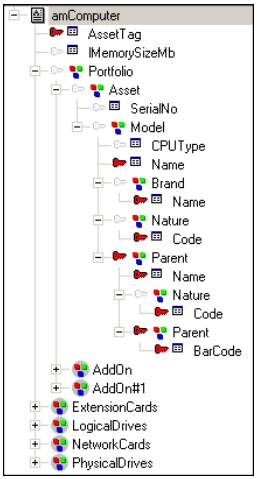
Enterprise DiscoveryからAssetCenterへのシナリオ - インベントリデータの照合更新

本シナリオを使用して、インベントリデータをAssetCenterアプリケーション向けに照合更新します。

本シナリオは、コンピュータテーブルと、ポートフォリオ品目、モデル、属性などのリンクされたテーブルにレコードを作成するのに使用するいくつかのマッピングから構成されています。

本シナリオのマッピングの構造は、edac.scnシナリオのものと類似しています。

以下で説明するマッピングは*DevicesSrc-amComputerDst*マッピングです。



注意:

AssetCenter version 4.4用に、FullNameが/*Network*/*Network Device Component*/であるモデルを作成します。このモデルは、*Network hardware*属性（コード*NET*）に関連付けられています。

*lMemorySizeMb*要素に関する照合更新提案が行われます。

照合更新提案

本シナリオでは、AssetCenterで利用可能な照合更新提案を、*IMemorySizeMb*要素のスクリプトから作成します。

照合更新スクリプトを以下に挙げます。

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

このスクリプトは、インベントリによって取得された値を、AssetCenterデータベースに存在する値とを比較します。

- 新しい値が従来の値より大きい場合、新しい値がデータベースに挿入されます。
- 新しい値がデータベースの値より小さい場合、照合更新提案がデータベース管理者に与えられ、データベース管理者はコンピュータメモリが減少したかどうかを検証できます。Connect-Itドキュメントのステータスは、*Pending*になります。

特別に設計された関数を呼び出すことで、保留レコード検証を照合更新提案テーブルに作成できる、*ValidateReconcUpdate* BASIC関数をこのスクリプトは使用します。

この関数は、識別子を検証することで照合更新提案が既に存在するかどうかを確認します。

注意:

この関数は、更新スクリプト（Update scriptフレーム）と併用する必要があります。

- 照合更新提案が存在しない場合、照合更新提案が作成され、レコードのステータスが*To assign*に設定されます。
- 照合更新提案が既に存在する場合、以下に挙げるようにそのステータスに応じて操作が変化します。
  - *Validated*の場合、要素の新しい値が対応テーブルに挿入されます。
  - *Rejected*の場合、ドキュメントは処理されません。AssetCenterでは更新は実行されません。
  - その他すべてのステータスの場合、レコードのステータスは*To assign*です。

```
'---------------------------------------------------------------------------
' strCode  = 更新提案の一意な識別子
' strName  = 更新提案の説明
' strTable = 変更に関わるテーブル
' strField = 変更された値を持つフィールド
' vNew     = フィールドの新しい値
' vOld     = フィールドの以前の値
```

```
' lRecId   = 変更されたフィールドのメインID
'-------------------------------------------------------------------------
Function ValidateReconcUpdate(ByVal strCode As String, ByVal strName As
String, ByVal strTable As String, ByVal strField As String, ByVal vNew As V
ariant, ByVal vOld As Variant, ByVal lRecId As Long) As Variant
ValidateReconcUpdate = CheckReconcProposal(strCode, strName, strTable, s
trField, vNew, vOld, lRecId)
End Function
```

このシナリオを再開すると、Connect-Itは照合更新提案のステータスを確認します。データベース管理者が照合更新提案を検証している場合、コンピュータテーブルにある対応レコードが更新されます。

注意:

照合更新提案が検証され、更新に使用される値が照合更新提案で検証された値の1つである場合、Connect-Itがレコードを更新します。この値は、Connect-Itで入力した値、データベースにあるこれまでの値、または検証された別の値とすることができます。

索引

A

F