Peregrine

Connect-It
ユーザガイド

PART NUMBER CIT-3.2.0-JAP-01001-00258

Peregrine Systems, Inc.
Worldwide Corporate Campus and Executive Briefing Center
3611 Valley Centre Drive San Diego, CA 92130
Tel 800.638.5231 or 858.481.5000
Fax 858.481.1751
www.peregrine.com

目次

図の一覧表

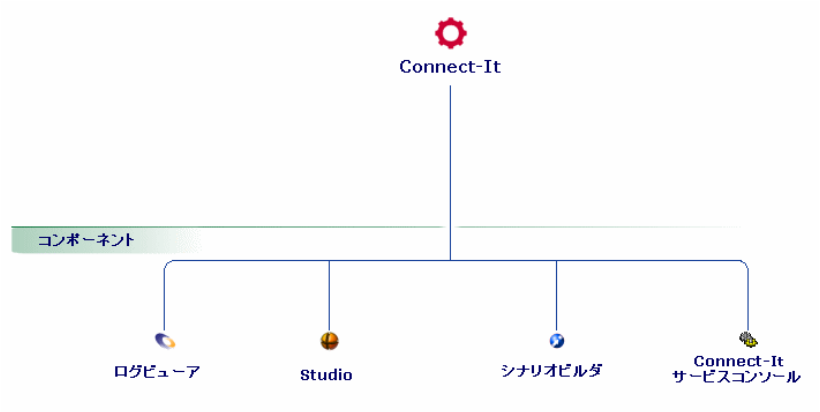
表の一覧表

序文

はじめに

Peregrine SystemアプリケーションであるConnect-Itは、以下の図にあるコンポーネントから構成されています。

図 1. Connect-It - アプリケーションのコンポーネント



Connect-Itの使用目的

Connect-Itは、EAI（Enterprise Application Integration）グループの一部を成す統合プラットフォームです。データを企業の内部や外部で取得または提供する様々なアプリケーション（内部用は備品管理ソフトウェア、内部技術サポートやLDAPディレクトリ。外部用はERP、B2B、B2C）を、EAIにより統合することができます。

Connect-Itは、データだけではなく企業アプリケーションのプロセスも統合します。

Connect-Itには以下の用途があります。

- あるデータベースから別のデータベースへデータを転送する。
- 2つの異種のデータベース間でデータを複製する。
- Eメール、区切り文字で区切られたテキストファイル、XMLファイルやその他のフォーマットのデータを、データベースにインポートする。
- データベースから、Eメール、区切り文字で区切られたテキストファイル、XMLファイルやその他のフォーマットへデータをエキスポートする。
- NTセキュリティ情報をデータベースにインポートする。
- その他

Connect-Itの対象ユーザ

Connect-Itは、企業内の異種のアプリケーション間の統合を担当するIT技術者を対象としています。

Connect-Itを使用するには以下の知識が必要になります。

- 統合されるアプリケーションに関する高度な知識
- マッピングスクリプトで使用されるBasic言語の知識
- JCA（Java Connector Architecture）規格の知識（Java開発キットを使ってコネクタを作成する場合）

本書の使用方法

「基本概念」の章

この章では、Connect-It機能の概要が説明されています。

「インストール」の章

この章では、Connect-Itのインストール方法が説明されています。

「シナリオビルダ」の章

この章では、統合シナリオ作成用のシナリオビルダのユーザインタフェースが説明されています。

「統合シナリオのインプリメンテーション」の章

この章では、統合シナリオのインプリメンテーション方法が説明されています。インプリメンテーションでは以下の操作を実行します。

- ソースコネクタとターゲットコネクタを選択する

  コネクタ（データベース型コネクタ、Eメールコネクタなど）は外部アプリケーションと通信し、Connect-Itシナリオがこれらの外部アプリケーションを統合します。
- コネクタの生成用ドキュメントタイプと、取り込み用ドキュメントタイプを作成する

  外部アプリケーションが生成するデータ全体（データベーステーブル、Eメール、テキストファイル、プロパティフォーマットなど）をXML形式に変換したものが、ドキュメントタイプに当たります。

このインプリメンテーションに関する情報は、次章で説明されるマッピングの手順で補足されます。

「ドキュメントタイプのマッピング」の章

この章では、ソースコネクタの生成用ドキュメントタイプと、ターゲットコネクタの取り込み用ドキュメントタイプの間のマッピングが、説明されています。

マッピングは、ソースコネクタの生成用ドキュメントタイプのデータを変換し、ターゲットコネクタがデータを取り込めるようにします。

「ピボットドキュメントタイプ」の章

この章では、ピボットドキュメントタイプが説明されています。ピボットドキュメントタイプは、マッピングなしで、ソースコネクタとターゲットコネクタ間でデータを転送できるようにします。

「統合シナリオのテストとデバッグ」の章

この章では、プロダクションモードで統合シナリオを使用する前に、テストとデバッグを行う方法が説明されています。

「シナリオ文書」の章

この章では、統合シナリオのシナリオ文書を動的に作成する機能が説明されています。このシナリオ文書には、シナリオで使用されるコネクタ、ソースドキュメントタイプとターゲットドキュメントタイプ、マッピングの詳細と、その他多数の情報が含まれています。

「統合シナリオをプロダクションモードにする」の章

この章では、統合シナリオをプロダクションモードにする方法が説明されています。シナリオのスケジュール（シナリオ実行の頻度）、Windowsのサービスとしてシナリオを使用する方法、Unixのデーモンとしてシナリオを使用する方法が解説されています。

「処理レポート」の章

この章では、コネクタが使用する処理レポートが説明されています。このレポートは、コネクタがデータを処理した方法を記述するドキュメントタイプです。データ処理の成功または失敗の情報は、他のコネクタのアクションをトリガするために使用されることもあります。アクションには、処理の停止、ヘルプデスクへの警告メッセージの送信などがあります。

「Connect-ItのJava開発キット（JDK）」の章

この章では、Connect-It付属のJava開発キットが説明されています。この開発キットを使うと、JCA（Java Connector Architecture）規格に基づいてJavaコネクタを作成できるようになります。

「問題点の報告方法」の章

この章では、Pegrine Systemsヘルプデスクに問題を報告する方法が説明されています。Peregrine Systemsの全製品に共通する情報に加えて、「Connect-Itに関する問題を報告する場合 [p. 255]」の節では、Connect-It用の報告方法が解説されています。

「用語解説」の章

この用語解説では、Connect-It特有の専門用語の一部が説明されています。

1 基本概念

データ処理

図 1.1. Connect-Itで、あるデータベースから別のデータベースへデータを転送する

Connect-Itは、コネクタを使用して外部アプリケーションと相互に作用します。あるデータベースから別のデータベースへデータを転送する場合、

- ソースコネクタはXML形式のドキュメントを生成します。

  生成される各ドキュメントは、ソースアプリケーション内の1データ集合に一致します。
- マッピングボックスが、ソースコネクタにより生成されたドキュメントの構造を再構築し、ターゲットコネクタがドキュメントを取り込めるようにします。
- ターゲットコネクタがXML形式のドキュメントを取り込みます。

  取り込まれた各ドキュメントは、ターゲットアプリケーション内の1データ集合に一致します。

データベース型のコネクタでは、各ドキュメントは1データベーステーブルのレコード（または別のテーブルへのリンク）に対応しています。例：AssetCenterデータベースでは、1ドキュメントは資産のテーブルのデータを含みます。

他のコネクタの場合、データ集合は、区切り文字で区切られたテキスト、Eメールメッセージやセキュリティ情報などに当たります。

コネクタの選択、コネクタ間の関係、そしてソースドキュメントとターゲットドキュメント間のマッピングの定義が、統合シナリオを構成します。テストとデバッグを経た後、シナリオはスケジューラに関連付けられ生産段階に入ります。

どの統合過程でも起こるように、ドキュメントの一部または全体が拒否されることもあります。しかし処理レポートやドキュメントログを使用すると、統合シナリオ全体を定義し直すことなく、拒否されたドキュメントを再処理することができます。

2 インストール

本章ではConnect-It統合プラットフォームのインストール方法を説明します。

サポートされる動作環境

Connect-ItはWindowsと互換性があります。

Connect-ItをUnix上（Linux、Solaris）にインストールすることは可能ですが、この場合Connect-Itは非グラフィックモードでしか機能しません。

Connect-Itは、使用予定のシナリオに関連するアプリケーションにアクセスできるコンピュータ上に、インストールされなければなりません。一般的に、シナリオに関連するアプリケーションのクライアント部分を、コンピュータに完全インストールする必要はありません。しかし最初のテスト段階などでは、外部アプリケーション内へ書き込まれるデータ結果を確認するために、コンピュータにアプリケーションをインストールしておくと便利です。

ドキュメントのトラッキング管理では、以下の要素をハードディスクに保存します。

- ドキュメント（「.dat」ファイル）
- トラッキング項目とその内容を説明するメッセージ（「.msg」ファイル）
- 迅速なアクセスを可能にするドキュメントの完全なインデックス（「.idx」ファイル）

ログに割り当てられたメモリ容量は、シナリオビルダの［**編集**／**オプション**］メニューのコマンドで指定できます。ログに割り当てられたデフォルトのメモリ容量は5MBです。

ドキュメントログとドキュメントのトラッキング管理についての詳細は、「シナリオビルダ [p. 27]」の章の「ログ [p. 49]」の節、「ドキュメントログ [p. 51]」を参照してください。

Connect-Itのインストール

Connect-Itは、Windows 32ビットまたはUNIX上にインストール可能です。

Connect-ItをWindows 32ビット上でインストールする

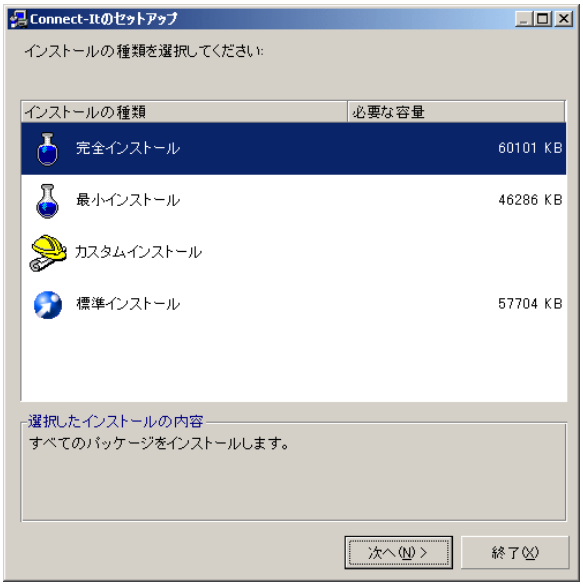
1. 必要な稼働環境があるか確認します。
2. Windowsを管理者の権限で起動します。

CD ROMを挿入すると、インストールプログラムが自動的に起動します。

自動的に起動しない場合はConnect-Itインストール先フォルダを開き、「setup.exe」プログラムを起動します。

インストールプログラムでは以下の4種のインストールから選択できます。

- 完全インストール
- 最小インストール
- カスタムインストール
- 標準インストール



最小インストール

最小インストールには以下のコンポーネントが含まれています。

- シナリオビルダ（Connect-It）
- Connect-Itコンソール。これによりシナリオに関連したサービスを作成できます。
- LOGファイルを読み取るためのログビューア
- ベースコネクタ
- Connect-Itデータキットには以下の要素が含まれています。
  - インストールされるコネクタに関連する付属シナリオ
  - 付属シナリオの使用に必要な関連ファイル

標準インストール

標準インストールには、最小インストールに加えて、PDFとCHMフォーマットのConnect-Itマニュアルが含まれます。

完全インストール

完全インストールには、標準インストールに加えて、以下のコンポーネントが含まれています。

- オプションコネクタ [p. 23]
  これは認証証明書が許可している場合に使用可能です。

- オプションコネクタに関連する付属シナリオ

カスタムインストール

カスタムインストールでは最小インストールを実行した上で、必要なコネクタを選択し追加することができます。

オプションコンポーネントをインストールすると、コンポーネントに関連する付属シナリオもインストールされます。

Connect-ItをUNIX上でインストールする

Connect-ItをUNIX上にインストールするには、以下のコマンドラインを実行して付属の「.tgz」ファイルを圧縮解除します。

```
tar xvfz cnit-[オペレーティングシステム]_[バージョン].[ビルド]_[言語の接頭文字].tgz
```

使用可能な接頭文字は、フランス語版では「fr」、英語版では「en」、イタリア語版では「it」、スペイン語版では「es」、ドイツ語版では「de」です。

UNIX上のConnect-Itのインストールは、常に完全インストールになっています。このインストールには以下のコンポーネントが含まれています。

- ベースコネクタ [p. 23]
- オプションコネクタ [p. 23]
- データキット [p. 24]
- 文書
- 用例シナリオ

注意:

コネクタの中には、Windows 32ビット環境でしか機能しないものもあります。コネクタの互換性に関する詳細情報については、ペレグリンシステムズのサポート用Webページhttp://support.peregrine.comを参照してください。

UNIX下のダイナミックライブラリ（.so）

Connect-Itでは、Connect-Itのインストール先フォルダの「bin」フォルダにあるダイナミックライブラリ（.so）を使用します。パス[Connect-Itのインストール先フォルダ]/binを、環境変数**LD_LIBRARY_PATH**に追加する必要があります。

コマンドC（csh）のインタープリタには、以下のコマンドラインを実行します。

```
setenv LD_LIBRARY_PATH=[Connect-Itのインストール先フォルダ]/bin
```

コマンドK（ksh）やBourne（sh）のインタープリタには、「.profile」ファイル内で以下のコマンドラインを実行します。

```
LD_LIBRARY_PATH=[Connect-Itのインストール先フォルダ]/bin export LD_LIBRARY_PATH
```

認証証明書を入力する

Connect-Itのインストールが終了した後、ペレグリンシステムズから入手した認証証明書をシナリオビルダ内に入力する必要があります。

認証証明書は以下の内容を含むテキストファイルです。

- 使用が許可されているオプションコネクタのリスト
- 各オプションコネクタの有効期限
- 暗号化された認証キー

認証証明書の入力方法

1. Connect-Itのシナリオビルダを起動します。
2. **［管理／認証証明書の編集］**メニューを選択します。
3. 表示されるダイアログボックスに認証証明書を入力します。
4. **［OK］**をクリックします。

   このアクションは、Connect-Itインストール先フォルダ内にlicense.txtを作成します。

注意:

UnixバージョンのConnect-Itにはグラフィカルインタフェースがありません。認証証明書を入力するには、以下の手順に従います。

1. ペレグリンシステムズから提供されるライセンスファイルの内容をコピーします。このファイルの内容は変更しないでください。
2. このファイルに「license.txt」と名前を付けてから保存します。
3. 「license.txt」をConnect-Itのインストール先フォルダにコピーします。

インストール内容

本節では、インストールされるコンポーネントとファイルについて説明します。

これらのファイルがコンピュータにインストールされるかどうかは、選択したインストールの種類（最小インストール、完全インストール、カスタムインストール）によります。

インストール先フォルダのファイル

Connect-Itインストール先フォルダのファイルの構成は、以下の表の通りです。

フォルダ名	主なファイル
bin32	• 実行可能ファイル（conitgui.exe、conitsvc.exe、console.exe、logview.exe） • ダイナミックライブラリ（DLL） • Connect-It専用のテキストエディタの設定ファイル（codeedit.cfg） • 付属シナリオとConnect-Itの機能に必要な多種のファイル
config	全シナリオ内のコネクタに関連したファイル • .mpt（マップテーブル） • .str（文字列テーブル） • .bas（Basic関数とグローバル変数）
datakit	Connect-Itツールで使用されるデータキット
doc	「.pdf」と「.chm」フォーマットの文書
fsf	「.fsf」ファイルのバックアップコピーの保存用フォルダ
scenario	• Connect-Itの用例シナリオを含んだSCNファイル • 用例シナリオ用のコネクタに関連したファイル
lib	JARファイルには次の内容が含まれています。 • Peregrine Systemsに開発されたJavaクラス • Javaサードパーティクラス（共有クラス）
wizards	Connect-Itの機能に必要なファイル
Studio	Connect-ItのStudio機能に必要なファイル

ベースコネクタ

ベースコネクタは、Connect-Itのインストールモードに関わらず必ずインストールされるコネクタです。

ベースコネクタは以下の通りです。

- Asset Managementコネクタ
- Action Request Systemコネクタ
- コマンドラインコネクタ
- InfraTools Desktop Discoveryコネクタ
- InfraTools Network Discoveryコネクタ
- Peregrine Desktop Inventoryコネクタ
- Desktop Administrationコネクタ
- InfraTools Managementコネクタ
- NTセキュリティコネクタ
- ゲートウェイ3.xコネクタ
- ServiceCenterコネクタ
- テキストコネクタ
- XMLコネクタ

オプションコネクタ

オプションコネクタは、Connect-Itの完全インストールを選択した場合、またはカスタムインストールでオプションコネクタを選択した場合にインストールされます。

オプションコネクタは以下の通りです。

- データベースコネクタ
- Eメールコネクタ
- XMLリスニングコネクタ
- Intel LANDeskコネクタ
- LDAPコネクタ
- Lotus Notesコネクタ
- MQSeriesコネクタ
- SCAutoリスニングコネクタ
- SMS 1.xコネクタとSMS 2.xコネクタ
- Tivoli Enterprise Consoleコネクタ（送信）
- Tivoli Enterprise Consoleコネクタ（受信）
- Tivoli Inventoryコネクタ（バージョン3.1と3.6）
- Tivoli Inventory CM for Inventory 4.2コネクタ

- Unicenter AMOコネクタ
- Webサービスコネクタ
- JDBCコネクタ
- SAP BAPIコネクタ
- SAP IDocコネクタ

追加コネクタ

追加コネクタは、デフォルトではシナリオビルダと共にインストールされません。これは、ペレグリンシステムズから提供されたCD-ROMを用いて、別々にインストールする必要があります。各追加コネクタ用にペレグリンシステムズから認証証明書が発行されます。

データキット

次の表はインストールされるデータキットの内容を説明しています。

データキット

ファイル名	機能
iddフォルダ	
*.fsf	Desktop Discoveryによるコンピュータのスキャンの例です。スキャンにはコンピュータ自体（Hardware）、コンピュータにインストールされたソフトウェア（Sotware）とコンピュータに関連する資産とユーザ（UserAndAssets）についての情報が含まれています。
master.sai、french.sai	このファイルにより、Desktop Discoveryは大多数のアプリケーションの情報を検索できます。ファイルには、各アプリケーションごとにバージョン、ソフトウェア会社、ライセンス番号などの情報が含まれています。この情報がない場合は、「.fsf」ファイル内にソフトウェアのファイル名だけが列挙されます。
iddsdt.cdt	このファイルは、Desktop Discoveryスキャナ生成プログラム（Scanner Generator）により生成されたデフォルトのスキャナに対応するドキュメントタイプを含んでいます。

データキット	
datakit.cdt	このファイルはサンプルのスキャン（インベントリ）に対応するドキュメントタイプを含んでいます。
scac36とscac41フォルダ	
modelfeat.scr modelfeat.txt	このスクリプトファイルにより、Asset Managementアプリケーション内で任意管理項目を作成できるようになっています。任意管理項目は、ServiceCenter-AssetCenter複製シナリオ内のServiceCenterの標準フィールドに関連付けられています。
acフォルダ	
stdfeatサブフォルダ	スクリプトファイル（sdtfeat.scr）をインポートすると、Asset Managementアプリケーション内の任意管理項目テーブル内にレコードが作成されます。これらの任意管理項目は、Asset Managementコネクタが発行する使用可能なドキュメントタイプ内に現れます。
tcフォルダ	このフォルダにあるファイルにより、TeleCenterコネクタを含む付属シナリオが使用可能になります。
indフォルダ	
acサブフォルダ	このフォルダは、Asset Managementアプリケーション内でカテゴリと任意管理項目を作成するインポートスクリプトを含んでいます。
doctransフォルダ	このフォルダは、シナリオ文書の自動生成に必要なファイルを含んでいます。

関連したファイル

数種のファイル（「.mpt」、「.str」、「.bas」ファイル）は、シナリオに関連付けられています。シナリオを別のフォルダへ移す場合は、これらのファイルもフォルダと共に移す必要があります。移動先では、シナリオファイルとの移動前の位置関係を再現しなければなりません。例：複数の用例シナリオに共通のファイルは「shared」フォルダにあります。このフォルダはシナリオの移動先にも作成されなければなりません。

注意:

ファイルマネージャを使用してシナリオを移動させるよりも、可能な限りシナリオビルダ内で直接移動先に保存することをお勧めします。これにより、シナリオに関連した全ファイルは、移動後も適切に参照されるようになります。

Connect-Itに関する情報

Connect-Itでは、ソフトウェアと動作環境に関する情報にアクセスできます。

情報にアクセスするには、

- **［ヘルプ／バージョン情報］**メニューを選択します。
- **［詳細］**をクリックします。

  以下の情報が表示されます。
  - ソフトウェア
  - システム情報
  - データベースエンジンに関する情報

3 シナリオビルダ

シナリオビルダは 統合シナリオを作成するためのグラフィカルインタフェースです。本章ではシナリオビルダの主要部分と、Connect-Itのオプションウィンドウについて説明します。

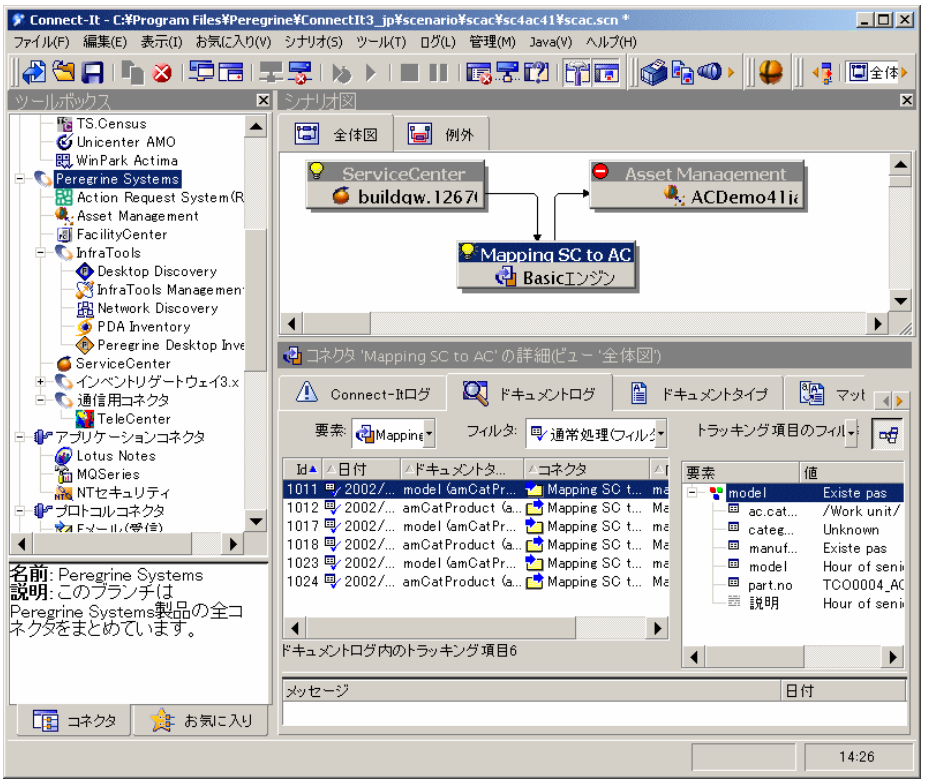
メインウィンドウ

シナリオビルダのメインウィンドウは以下の要素で構成されています。

- メニューバー

  メニューの詳細については、本章の「メニュー [p. 35]」の節を参照してください。
- ツールバー
- パネル（枠）

図 3.1. シナリオビルダ - メインウィンドウ



ツールバー

頻繁に行う操作は、シナリオビルダのメニューバーを使用せずにツールバーで実行できます。使用可能なツールバーは以下の通りです。

- **一般ツールバー**
- **シナリオ図のバー**
- **ビューのバー**
- **お気に入りのバー**
- **Studioバー**

表 3.1. シナリオビルダ - 一般ツールバーのアイコン

アイコン	機能
	ウィザードを起動し、1つのソースコネクタと1つのターゲットコネクタを使った最小シナリオを作成できるようにします。
	既存シナリオを開きます。
	シナリオを保存します。
	Connect-Itのテキスト要素をコピーします。 • ツールボックス内のコネクタの名前 • ログ内のメッセージ • その他
	シナリオ内で選択したコンポーネントを削除します。
	選択したコンポーネントの設定ウィザードが起動します。
	コンポーネントの設定ウィザードで、高度な設定ページを表示します。（例：**［ピボットドキュメントタイプを使用する］**のページ）
	シナリオの全てのコネクタを開きます。
	シナリオの全てのコネクタを閉じます。
	シナリオをスケジュールモードで起動します。
	シナリオをテストします（非スケジュールモード)。
	シナリオを停止します。
	シナリオを一時停止します。
	**［Connect-Itログ］**タブと**［ドキュメントログ］**タブ内の項目を削除します。
	作業のオフライン／オンラインモードを変更します。
	ターゲットコネクタを外部アプリケーションと相互作用させずに、シナリオをテストします。 トランザクションをサポートするデータベース型のコネクタでは、エラーメッセージが表示されます。エラーメッセージは、このオプションを使用しない場合に取得されるメッセージに相当します。

アイコン	機能
	**［ツールボックス］**枠を表示／非表示にします。
	**［シナリオ図］**枠を表示／非表示にします。

表 3.2. シナリオビルダ - お気に入りのツールバーのアイコン

アイコン	機能
	［ツールボックス］の**［お気に入り］**タブ内で選択したお気に入りを再設定できるようにします。
	［ツールボックス］の**［お気に入り］**タブ内で選択したお気に入りを削除します。
	「Favorite.fav」ファイルを、あるフォルダにインポートします。
	「Favorite.fav」ファイルを、あるフォルダにエキスポートします。 これらのお気に入りを別のユーザが使用できるようにするには、Connect-Itのインストール先フォルダの「bin32」フォルダ内に、このファイルをインポートします。

表 3.3. シナリオビルダ - シナリオ図のツールバーのアイコン

アイコン	機能
	シナリオ図を印刷します。 （印刷前のプレビュー用ウィンドウで、印刷をパラメータ設定できます。）
	シナリオ図をクリップボードにコピーします。
	このアイコンの横のレバーでシナリオ図を縮小／拡大します。

表 3.4. シナリオビルダ - ビューのバーのアイコン

アイコン	機能
	マッピングを作成する前にこのアイコンでビューを選択すると、マッピング作成後に、マッピングのソースコネクタとターゲットコネクタがビュー内に表示されます。

表 3.5. シナリオビルダ - Studioツールバーのアイコン

アイコン	機能
	現在のシナリオとStudioのプロジェクトの同期を取ります。 この機能の詳細については、Studioのマニュアルを参照してください。

ツールバーを表示／非表示にする

1. **［表示］**メニューを選択します。
2. 以下のツールバーを必要に応じて選択または非選択にします。
   - ツールバー（一般）
   - （シナリオ）図のバー
   - Studioバー
   - ビューのバー

ツールバーを移動させる

1. ツールバーの取っ手アイコン（  ）にマウスポインタを置きます。
2. この取っ手をクリックすると、ツールバーの周囲を囲む黒線の長方形が表示されます。
3. マウスボタンを押したままにして、希望の位置に黒線の長方形を移動させます。
4. ツールバーを囲む黒線の長方形が希望の位置に移ったら、マウスボタンを放します。

   ツールバーは、以下の位置に配置できます。
   - シナリオビルダの4つの隅
   - 画面上の全領域（フロート状態で）

フロート状態のツールバーを画面の端に固定する

1 ポインタを、フロート状態のツールバーの端に置きます。
2 ポインタの形状が変化するのを待ちます（↖↘）。
3 ダブルクリックします。
ツールバーは、自動的に元の場所に戻ります。

枠

シナリオビルダのメインウィンドウは、サイズ変更可能な3つの部分から成っています。

- シナリオのコンポーネントやお気に入りを選択するためのツールボックス
- シナリオ内のコンポーネントとコンポーネント間のリンクがグラフィックで表示されるシナリオ図部分
- 以下のタブを含む詳細画面
  - Connect-Itログのタブ
  - ドキュメントログのタブ
  - シナリオ図内で選択したコンポーネントに関連するタブ：生成用、または取り込み用ドキュメントタイプ、マッピング、ピボットマッピング。

枠を表示／非表示にする

以下の枠を表示／非表示にすることが可能です。

- ツールボックス枠
  この枠を表示／非表示にするには、以下のアクションのいずれかを実行します。
  - **［表示／ツールボックス］**を選択または非選択にします。
  - をクリックします。
- シナリオ図の枠
  この枠を表示／非表示にするには、以下のアクションのいずれかを実行します。
  - **［表示／シナリオ図］**を選択または非選択にします。
  - をクリックします。

枠を移動させる

1 枠の上端にマウスポインタを置きます。
2 濃淡のバーをクリックすると、枠の周囲を囲む黒線の長方形が表示されます。

3 マウスボタンを押したままにして、希望の位置に黒線の長方形を移動させます。
4 枠を囲む黒線の長方形が希望の位置に移ったら、マウスボタンを放します。

シナリオ図のビュー

［シナリオ図］枠には、2つのタブ**［全体図］**と**［例外］**があります。［全体図］タブは、シナリオの全コンポーネント（ソースコネクタ、ターゲットコネクタ、リンクとマッピングボックス）を表示します。［例外］タブにはデフォルトでは何も表示されませんが、このタブはシナリオのエラーの管理用に用意されています。

［シナリオ図］枠内でビューを使用すると、一部のマッピングで使用されるコンポーネントのみを表示できるようになります。

例：複数のソースコネクタを使用するシナリオで、1つのソースコネクタのみを含むマッピングを選択すると、このソースコネクタのみが表示されるようになります。

シナリオ図のビューを使用する

あるシナリオ用にビューを作成すると、このシナリオをシナリオビルダ内で開くたびに、ビューが表示されます。

シナリオ図のビューの作成方法

1 シナリオを開きます。
2 **［シナリオ／ビューの管理］**を選択します。
3 表示されるウィンドウで、をクリックします。
4 ビューに名前を付けます。ビューの名前はシナリオ内で固有の名前でなければなりません。
5 シナリオ図のビューで表示するマッピングを選択します。
6 **［作成］**をクリックします。

ビューエディタを閉じると、**［シナリオ図］**枠内にビューの名前がついたタブが表示されます。

シナリオ図のビューの変更方法

1 シナリオを開きます。
2 **［シナリオ／ビューの管理］**を選択します。
3 ビューを選択します。
4 ビューの名前やマッピングの選択内容を変更します。
5 **［変更］**をクリックします。

6 ［閉じる］をクリックします。

シナリオ図のビューの削除方法

1 シナリオを開きます。
2 ［シナリオ／ビューの管理］を選択します。
3 ビューを選択します。
4 をクリックします。
このビュー用のタブが［シナリオ図］枠に表示されなくなります。

シナリオ図のビューを印刷する

シナリオ図の各ビューを印刷することが可能です。

シナリオ図のビューを印刷する

1 ビューを選択します。
例：［全体図］
2 以下の操作の内1つを実行します。
- をクリックします。
- ［ファイル／シナリオ図を印刷する］を選択します。

3 印刷プレビュー用ウィンドウが表示されます。
4 ［印刷］をクリックします。

ビューの印刷を設定する

1 ビューの印刷用メニューを選択します。
2 ［設定］をクリックします。
3 Windowsの印刷設定用ウィンドウが表示されます。
4 設定パラメータを変更します。

ページ設定を変更する

1 ビューの印刷用メニューを選択します。
2 ［ページ設定］をクリックします。
3 ［ページ設定］ダイアログボックスが表示されます。
4 ページ設定の以下のパラメータを変更します。
- 左側の余白
- 上側の余白
- 右側の余白
- 下側の余白

- ヘッダーの余白
- フッターの余白
- ズーム

使用される単位はセンチメートルです。

- ページ番号（［#］）
- 現在の日付（［日付］）
- 現在の時刻（［時刻］）
- 「.scn」ファイルの完全パス

［ヘッダー］と［フッター］ボタンを押すと、別のダイアログボックスが表示されます。ここでヘッダーとフッターに印刷する以下の情報を指定できます。

［A］ボタンを押すと、印刷する情報の文字のフォントを選択できます。

シナリオ図のビュー - 使用上の規則

シナリオ図のビューは以下の規則に従います。

- ［全体図］以外のビュー内のコンポーネントを、ドラッグアンドドロップで移動させることはできません。
- ビュー内の1つのマッピングを削除しても、対応するSCNファイルの<LAYER>要素はなくなりません。
- マッピング、コネクタとリンクの編集は、［全体図］ビューのみで実行可能です。
- ビュー内のコンポーネントをクリックすると、ドキュメントログ内に関連するトラッキング項目のみが表示されるように、フィルタを設定できます。
- ビューは、Connect-Itの他の機能には影響しません。

メニュー

本節では以下の内容について説明します。

- シナリオビルダのメニューバー
- シナリオビルダのショートカットメニュー

シナリオビルダのメニューバー

［ファイル］メニュー

［ファイル］メニューのコマンドのリストは以下の通りです。

表 3.6. ［ファイル］メニューのコマンドの機能

コマンド	機能
新規作成	［シナリオの設定］ウィザードを起動します。このウィザードでは、ソースコネクタとターゲットコネクタを設定し選択します。ウィザードは2つのコネクタをリンクするマッピングボックスを作成します。
開く	既に作成されたシナリオファイル（「.scn」ファイル）を開きます。
保存...	シナリオを保存します。
名前を付けて保存...	開いたシナリオの元の名前とは別の名前を付けて、シナリオを保存します。
シナリオの履歴	現在のシナリオ用の情報を入力するためのウィンドウを表示します。
シナリオ図を印刷する	シナリオ図の印刷用パラメータを入力するためのウィンドウを表示します。
シナリオ図をコピーする	シナリオ図の画像をクリップボードにコピーします。この画像をJasc Paint Shop ProやAdobe Photoshopなどのグラフィック用アプリケーション内に貼り付けることができます。
シナリオ文書を作成する	シナリオ文書をXML、DBKまたはHTMLフォーマットで作成できます。 この機能の使用方法については、「シナリオ文書 [p. 155]」章の「シナリオ文書の作成 [p. 161]」の節を参照してください。
HTMLシナリオ文書を表示する	現在のシナリオ文書をインターネットブラウザ内に表示します。 この機能の使用方法については、「シナリオ文書 [p. 155]」章の「シナリオ文書をHTMフォーマットで表示する [p. 158]」の節を参照してください。
シナリオ文書のプロパティ	この機能の詳細については、「シナリオ文書 [p. 155]」の章の「シナリオ文書のプロパティを編集する [p. 159]」の節を参照してください。 シナリオ文書のプロパティのウィンドウが起動します。
終了	シナリオビルダを終了します。

コマンド	機能
ファイルのリスト	最近編集したシナリオを表示します。リストの1要素をクリックすると、シナリオを直接開くことができます。

[編集] メニュー

[編集] メニューのコマンドのリストは以下の通りです。

表 3.7. [編集] メニューのコマンドの機能

コマンド	機能
切り取り	なし
コピー	Connect-Itログまたはドキュメントログの項目を、コンピュータのクリップボードにコピーします。
貼り付け	なし
[削除]	シナリオから選択したコンポーネントを削除します。
オプション...	シナリオビルダのオプションウィンドウを表示します。

[表示] メニュー

[表示] メニューのコマンドのリストは以下の通りです。

表 3.8. [表示] メニューのコマンドの機能

コマンド	機能
ツールボックス	コネクタのリストを含む枠を表示／非表示にします。
シナリオ図	シナリオ図を表示／非表示にします。
ツールバー	ツールバーを表示／非表示にします。
図のバー	シナリオ図用のツールバーを表示／非表示にします。
お気に入りのバー	お気に入りのツールバーを表示／非表示にします。

コマンド	機能
Studioバー	統合シナリオとStudioプロジェクトの同期をとるためのツールバーを表示／非表示にします。 この機能の詳細については、『Peregrine Studio 3.1.0 - User's Guide』を参照してください。

[シナリオ] メニュー

［シナリオ］メニューのコマンドのリストは以下の通りです。

表 3.9. ［シナリオ］メニューのコマンドの機能

コマンド	機能
全コネクタを開く	シナリオの全てのコネクタとマッピングボックスを開きます。
全コネクタを閉じる	シナリオの全てのコネクタとマッピングボックスを閉じます。
内部コネクタを表示する	ピボットドキュメントタイプを使用している場合に、Connect-It が自動的に作成する内部コネクタを表示します。
キャッシュの同期をとる	シナリオ内で開かれた全コネクタのキャッシュファイルの同期をとります。
キャッシュを削除する	シナリオ内で開かれた全コネクタのキャッシュファイルを削除します。
ビューの管理	ビューエディタを表示します。 ビュー管理の詳細については、本章の「メインウィンドウ [p. 27]/」の節の「シナリオ図のビュー [p. 33]」を参照してください。
スケジューラ	スケジューラの編集ウィンドウを表示します。
スケジュール	スケジュールの編集ウィンドウを表示します。
文字列テーブル	文字列テーブルの編集ウィンドウを表示します。
マップテーブル	マップテーブルの編集ウィンドウを表示します。
グローバル関数	グローバル関数と変数の編集ウィンドウを表示します。

コマンド	機能
ユーザフォーマット	マッピングスクリプトで使用される日付と数値のフォーマットを作成できます。
オフラインで作業する	オフラインで作業できるようにします。 ツールバーの をクリックすると、このオプションをオンにすることができます。 オフライン作業の詳細については、「統合シナリオのテストとデバッグ [p.149]」章の「オフラインで作業する [p.152]」の節を参照してください。
シナリオのテストモード	シナリオをテストする際に、ターゲットコネクタが外部アプリケーションにデータを転送しないように設定します。 ツールバーの クリックすると、このオプションをオンにすることができます。
全スケジューラを起動する	シナリオのソースコネクタの生成用ドキュメントタイプに関連するスケジューラを起動します。
中断	シナリオを停止します。
一時停止	シナリオの実行中に一時停止します。

[ツール] メニュー

［ツール］メニューのコマンドのリストは以下の通りです。

表 3.10. ［ツール］メニューのコマンドの機能

コマンド	機能
設定	選択したコンポーネントの設定ウィザードが起動します。
高度な設定	コンポーネントの設定ウィザードで高度な設定ページを表示します（［F2］キー）。 ツールバーの クリックすると、このオプションをオンにすることができます。
開く	選択したコンポーネントを開きます。
閉じる	選択したコンポーネントを閉じます。
キャッシュ／キャッシュの同期をとる	選択したコネクタのキャッシュファイルの同期をとります。
キャッシュ／キャッシュを削除する	選択したコネクタのキャッシュファイルを削除します。

コマンド	機能
ドキュメントタイプを編集する	**［ドキュメントタイプの選択］**ウィザードが起動します。選択したコンポーネントの生成用、または取り込み用ドキュメントタイプを選択します。
マッピングの編集	**［マッピングの選択］**ウィザードが起動します。選択したマッピングボックスのマッピングを選択できます。
生成する	ドキュメントタイプが作成されているコンポーネントのドキュメントを生成します。
Studioを起動する／Studioの同期をとる	Studioを起動し、シナリオと、Studioでシナリオに関連しているプロジェクトの同期をとります。 この機能の詳細については、『Peregrine Studio 3.1.0 - User's Guide』を参照してください。

［ログ］メニュー

［ログ］メニューのコマンドのリストは以下の通りです。

表 3.11. ［ログ］メニューのコマンドの機能

コマンド	機能
ドキュメントログを設定する...	ドキュメントログを設定するためのウィンドウを表示します。
ドキュメントログを更新する	ドキュメントログ内の内容を更新します。
ドキュメントログを再び読み込む	**［ドキュメントログ］**タブ内に、ドキュメントログファイルに保存されたトラッキング項目全体を表示します。
表示されたトラッキング項目を削除する	**［ドキュメントログ］**タブから、表示されているトラッキング項目全体を削除します。
格納されたトラッキング項目を削除する	ドキュメントログファイルに保存されているトラッキング項目を削除します。

［管理］メニュー

［管理］メニューのコマンドのリストは以下の通りです。

表 3.12. ［管理］メニューのコマンドの機能

コマンド	機能
認証証明書を編集する...	認証証明書の編集ウィンドウを表示します。

［Java］メニュー

［Java］メニューのコマンドのリストは以下の通りです。

表 3.13. ［Java］メニューのコマンドの機能

コマンド	機能
JMVを設定する	Javaクラスのパスを指定するためのウィンドウが表示されます。
コネクタを導入する	作成するJavaコネクタの導入用ウィンドウが表示されます。

［ヘルプ］メニュー

［ヘルプ］メニューのコマンドのリストは以下の通りです。

表 3.14. ［ヘルプ］メニューのコマンドの機能

コマンド	機能
オンラインヘルプ	Connect-Itのオンラインヘルプを表示します。
ご存知でしたか?	［ご存知でしたか?］ボックスを表示します。
バージョン情報...	使用中のConnect-Itのバージョンに関する一般的な情報を含む［バージョン情報...］ウィンドウを表示します。

シナリオビルダのメインウィンドウ内のショートカットメニュー

シナリオビルダのメインウィンドウ内では、以下の場合に右クリックメニュー（ショートカットメニュー）を表示します。

- マウスポインタがシナリオ図部分にある場合
- マウスポインタがConnect-Itログのタブ内にある場合
- マウスポインタがドキュメントログのタブ内にある場合

マウスポインタがシナリオ図部分にある場合

ポインタがシナリオ図部分にある場合に表示されるショートカットメニューのコマンドのリストは、以下の通りです。

コマンド	機能
コネクタを設定する...	選択したコンポーネントを設定するための**［コネクタの設定］**ウィザードが起動します。
コネクタを開く	選択したコンポーネントを開きます。
コネクタを閉じる	選択したコンポーネントを閉じます。
キャッシュ／キャッシュの同期をとる	選択したコネクタのキャッシュファイルの同期をとります。
キャッシュ／キャッシュを削除する	選択したコネクタのキャッシュファイルを削除します。
ドキュメントタイプを編集する	**［ドキュメントタイプの選択］**ウィザードが起動します。選択したコンポーネントの生成用、または取り込み用ドキュメントタイプを選択します。
マッピングを編集する...	**［マッピングの選択］**ウィザードが起動します。選択したマッピングボックスのマッピングを選択できます。
生成する	ドキュメントタイプが作成されているコンポーネントのドキュメントを生成します。
削除	選択したコンポーネントを削除します。
トラッキング項目の表示	選択したコンポーネントに関連するトラッキング項目をドキュメントログ内に表示します。
ツールボックスを表示する	ツールボックスを表示／非表示にします。
シナリオ図を表示する	シナリオ図を表示／非表示にします。

ポインタがConnect-Itログのタブ内にある場合

コマンド	機能
1レベル表示	トラッキング項目を1レベル分表示します。
全レベル表示	トラッキング項目を全レベル表示します。

コマンド	機能
全てのレベルを非表示	トラッキング項目の全てのレベルを非表示にします。
ツールボックスを表示する	ツールボックスを表示／非表示にします。
シナリオ図を表示する	シナリオ図を表示／非表示にします。

ポインタがドキュメントログのタブ内にある場合

ドキュメントログ内では2つのショートカットメニューがあります。

- トラッキング項目上にポインタがある場合
- ソースドキュメントの詳細の一要素上にポインタがある場合

トラッキング項目上にポインタがある場合

コマンド	機能
このXML文書をコピーする	コンピュータのクリップボードに、トラッキング項目に対応するXML文書をコピーします。
このXMLのDTDをコピーする	コンピュータのクリップボードに、トラッキング項目に対応するXMLのDTD（文書型定義）をコピーします。
このXML文書を開く	オペレーティングシステムで指定されている、XMLファイルに関連したアプリケーションでXML文書を開きます。（例：Internet Explorer）
ツールボックスを表示する	ツールボックスを表示／非表示にします。
シナリオ図を表示する	シナリオ図を表示／非表示にします。

ソースドキュメントの詳細の一要素上にポインタがある場合

コマンド	機能
このXML文書をコピーする	コンピュータのクリップボードにXML文書をコピーします。
このXMLのDTDをコピーする	コンピュータのクリップボードに、トラッキング項目に対応するXMLのDTD（文書型定義）をコピーします。

コマンド	機能
このXML文書を開く	オペレーティングシステムで指定されている、XMLファイルに関連したアプリケーションでXML文書を開きます。（例：Internet Explorer）
パスをコピーする	コンピュータのクリップボードに、選択した要素のパスをコピーします。
ツールボックスを表示する	ツールボックスを表示／非表示にします。
シナリオ図を表示する	シナリオ図を表示／非表示にします。

お気に入り

「お気に入り」とは既に設定されたコネクタのことを指します。これは、シナリオビルダのツールボックスの［**お気に入り**］タブ内に表示されます。

シナリオで（ドラッグアンドドロップして）お気に入りを使用すると、このお気に入りに関連付けられたコネクタのインスタンスが作成されます。後で、このコネクタのインスタンスとお気に入り間の関連付けを削除することもできます。

お気に入りが1つまたは複数のコネクタに関連付けられている場合、

- お気に入りを再設定すると、このお気に入りに関連付けられているコネクタは、すべてのシナリオ内で自動的に再設定されます。

  警告:

  お気に入りの名前と説明は、お気に入りでのみ使用されます。設定ウィザードでお気に入りの名前や説明を変更しても、関連付けられたコネクタの設定内容は変更されません。

  この設定更新を有効にするには、シナリオビルダを使って新たにシナリオを開く必要があります。
- 関連付けられたコネクタを再設定すると、このコネクタに関連付けられたお気に入りと他のコネクタが、再設定されることがあります。

  コネクタの再設定が終了すると、ダイアログボックスに以下の選択肢が表示されます。
  - 関連付けられたお気に入りを更新する
  - お気に入りとの関連付けを削除してから、コネクタの設定を更新する
  - コネクタとお気に入りの再設定をキャンセルする

お気に入りの編集

お気に入りを使うと、コネクタの設定を保存し、他のシナリオで再利用することが可能になります。

コネクタに関連付けされたお気に入りを作成する

1. シナリオビルダを起動します。
2. 既存シナリオを開くか、または新規にシナリオを作成します。
3. コネクタを選択して、どのコネクタ用にお気に入りを作成するかを指定します。
4. 右クリックします。
5. **［お気に入り／お気に入りに追加する（関連付けあり）］**をショートカットメニューから選択します。

   ［ツールボックス］の**［お気に入り］**タブ内にお気に入りが表示されます。

注意:

コネクタがお気に入りに関連付けられている場合、シナリオ図内のコネクタの上にお気に入りの名前が表示されます。

コネクタに関連付けられないお気に入りを作成する

1. シナリオビルダを起動します。
2. 既存シナリオを開くか、または新規にシナリオを作成します。
3. コネクタを選択して、どのコネクタ用にお気に入りを作成するかを指定します。
4. 右クリックします。
5. **［お気に入り／お気に入りに追加する（関連付けなし）］**をショートカットメニューから選択します。

   ［ツールボックス］の**［お気に入り］**タブ内にお気に入りが表示されます。

シナリオ内でお気に入りを使用する

1. シナリオを開くかまたは新規作成して、どのシナリオでお気に入りを使用するかを指定します。
2. **［ツールボックス］**の**［お気に入り］**タブ内にマウスポインタを置きます。
3. お気に入りを選択します。
4. 以下の操作のいずれかを実行します。
   - このお気に入りを、シナリオ図部分へドラッグアンドドロップします。
   - ダブルクリックします。

このお気に入りに自動的に関連付けられたコネクタのインスタンスが、シナリオ図内に作成されます。

コネクタとお気に入り間の関連付けを削除する

1. お気に入りに関連付けられたコネクタを、シナリオ内で選択します。
   コネクタがお気に入りに関連付けられていると、お気に入りの名前がシナリオ図内に表示されます。
2. 右クリックします。
3. **［お気に入り／お気に入りへの関連付けを削除する］**を選択します。
   お気に入りとの関連付けが削除されると、シナリオ図のコネクタの上に表示されていたお気に入りの名前が消えます。

［お気に入り］を使ってコネクタを再設定し、関連付けを作成する

1. お気に入りを選択します。
2. マウスボタンを押したままにして、お気に入りに類似したコネクタへお気に入りをドラッグします。
   例：XMLお気に入りをXMLコネクタへドラッグする
3. マウスボタンから指を離します。
4. 表示されるダイアログボックスで、お気に入りの設定でコネクタを更新し同時に関連付けを作成することを選択します。
5. **［OK］**をクリックします。

［お気に入り］を使ってコネクタを再設定するが、関連付けは作成しない

1. お気に入りを選択します。
2. 以下の方法で、お気に入りを類似するコネクタへドラッグします。
   - マウスの左ボタンを押したままにします。
   - **［Shift］**キーを押したままにします。
3. マウスボタンから指を離します。
4. 表示されるダイアログボックスで、関連付けを作成せずにお気に入りの設定でコネクタを更新することを選択します。
5. **［OK］**をクリックします。

［お気に入り］タブでお気に入りを再設定する

1. シナリオビルダを起動します。

2 **［ツールボックス］の［お気に入り］タブを選択します。**
3 **再設定するお気に入りを選択します。**
4 **以下の操作のいずれかを実行します。**
-  **をクリックします。**
- **右クリックして、ショートカットメニューから［お気に入りを設定する］を選択します。**
- **［お気に入り／お気に入りを設定する］を選択します。**

5 **起動する設定ウィザードで、お気に入りの設定を変更します。**

このお気に入りを使用するシナリオを読み込むと、関連付けられているコネクタは自動的に再設定されます。

関連付けられているコネクタでお気に入りを再設定する

1 **シナリオビルダを起動します。**
2 **再設定するお気に入りに関連付けられているコネクタを使用するシナリオを開きます。**
3 **関連付けられているコネクタを選択します。**
4 **以下の操作のいずれかを実行します。**
- **右クリックして、ショートカットメニューから［コネクタを設定する］を選択します。**
- **［ツール／設定］（［F2］キー）を選択します。**

5 **起動する設定ウィザードで、お気に入りの設定を変更します。**
6 **表示されるダイアログボックスで［はい］をクリックします。**

このダイアログボックスで［いいえ］をクリックすると、お気に入りとの関連付けが削除されてから、コネクタの設定が更新されます。

このお気に入りを使用するシナリオを読み込むと、関連付けられているコネクタは自動的に再設定されます。

お気に入りを削除する

1 **シナリオビルダを起動します。**
2 **［ツールボックス］の［お気に入り］タブを選択します。**
3 **削除するお気に入りを選択します。**
4 **以下の操作のいずれかを実行します。**
-  **をクリックします。**
- **右クリックして、ショートカットメニューから［お気に入りを削除する］を選択します。**
- **［お気に入り／お気に入りを削除する］を選択します。**

5 **［お気に入り］タブからお気に入りが削除されます。**

注意:

お気に入りが削除されると、関連付けられているコネクタは関連付けを失います。これにより、シナリオ図のコネクタの上に表示されていたお気に入りの名前も消えます。

お気に入りのファイルを管理する

お気に入り（既定のコネクタ）のパラメータは、「Favorite.fav」ファイル内に含まれています。このファイルは、Connect-Itのインストール先フォルダの「bin32」フォルダに保存されます。**例**：C:\Program Files\Peregrine\ConnectIt\bin32\Favorite.fav

注意:

「favorite.fav」ファイルの最終バージョンは、「favorite.bak」ファイル内に保存されます。更新時に問題が起こった場合は、このファイルを名前を変更して使用します。

このファイルを、2つのシナリオビルダ間でインポートまたはエキスポートすることも可能です。

シナリオファイルをインポートする

1. シナリオビルダを起動します。
2. **［お気に入り］**タブページ内にマウスポインタを置きます。
3. 以下の操作の内1つを実行します。
   -  をクリックします。
   - 右クリックして**［お気に入りをインポートする］**ショートカットメニューを選択します。
4. シナリオビルダ内にインポートするお気に入りを含む「Favorite.fav」ファイルのパスを入力します。

注意:

Windowsエクスプローラで「Favorite.fav」ファイルをコピーし、Connect-Itインストール先フォルダの「bin32」フォルダ内に貼り付けて、この操作を手動で実行することも可能です。

シナリオファイルを出力する

1 シナリオビルダを起動します。
2 以下の操作のいずれかを実行します。
   - ［**お気に入り**］タブページ内にマウスポインタを置きます。右クリックし、ショートカットメニューから［**お気に入りを出力する**］を選択します。
   -  をクリックします。
3 使用中のシナリオビルダのお気に入りを含む「Favorite.fav」ファイルを、どのフォルダに保存するかを指定します。

 **注意:**

Windowsエクスプローラで「Favorite.fav」ファイルをコピーし、別のフォルダ内に貼り付けて、この操作を手動で実行することも可能です。

関連付けられたお気に入りとコネクタを更新する

シナリオを開くと、お気に入りの更新日と関連付けられたコネクタの更新日に応じて、2コンポーネントの同期がとられます。

より新しい更新日時のお気に入りまたはコネクタの設定が、別のコンポーネントに適用され、同期がとられます。

ログ

シナリオビルダには2つのログがあります。

- Connect-Itログ
- ドキュメントログ

Connect-Itログ

ユーザはConnect-Itで実行された全アクションをログ内で確認できます。

例：コネクタが開いたことを知らせるメッセージが表示されます。

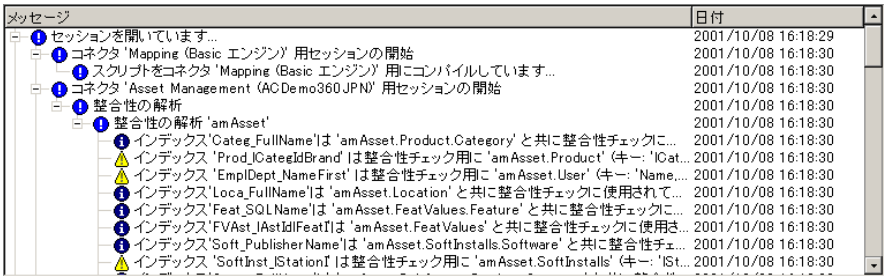
シナリオビルダの**Connect-Itログ**タブを選択すると、Connect-Itログにアクセスできます。

ログでは各アクションがアイコン  で表示されます。アクションのメッセージには、場合によってアクションの詳細を説明するサブメッセージがあります。サブメッセージには更にサブメッセージが付くこともあります。

各メッセージには、アクションが起動した日付が付きます。

ログ内で右クリックするとショートカットメニューが表示され、1レベル表示、全レベル表示または非表示を選択できます。

図 3.2. Connect-Itログ



Connect-Itログ内で取得されるメッセージの例については、マニュアル**『コネクタ』**の**「コネクタのルール（ディレクティブ）」**の章**「取り込み用ルール」**の節、**「整合性」**の**「識別キーに関するエラーメッセージ」**を参照してください。

Connect-Itログ内で使用されるアイコン

トラッキング項目	意味	メッセージのタイプ	例
	アクションについての情報	Connect-Itのアクションを説明します。	コネクタの開始
	アクションの詳細	アクションを詳細に説明します。	整合性チェックの段階で特定のインデックスが使用されている。
	警告	データの不適切な処理を発生させる問題点を報告します。	識別キーとして選択されたフィールドは重複を許可している（一意性がない）。
	重大な問題	アクションが失敗した理由を説明します。	コネクタの設定が無効である。外部アプリケーションとの通信が不可能。

ログから全メッセージを削除するには、**［ログ］**メニューの**［表示されたトラッキング項目を削除する］**オプションを選択するか、または をクリックします。

Connect-Itログを削除するとドキュメントログも削除されます。

ドキュメントログ

ドキュメントに以下の操作が実行された後、ドキュメントログでドキュメントの詳細を参照、確認できます。

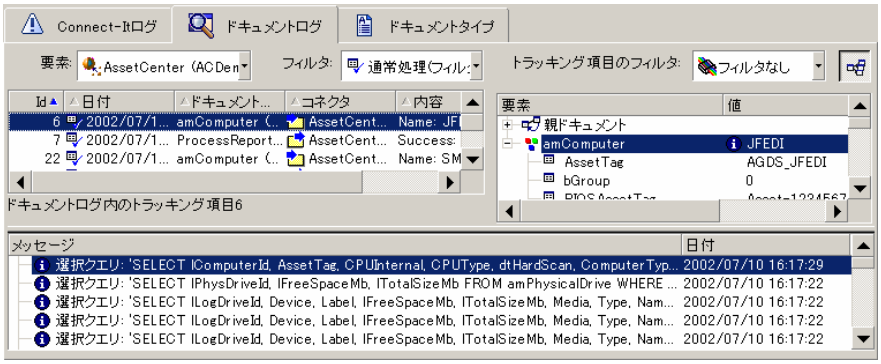
- ドキュメントがコンポーネントにより生成される、または取り込まれる。
- ドキュメントがリンクにより送信される。

シナリオビルダで［ドキュメントログ］タブを選択すると、ドキュメントログが表示されます。

このタブは3つの部分から成っています。

- 1ドキュメントにつき1つのトラッキング項目が表示される部分（左）
- 選択したドキュメントの詳細項目が表示される部分（右）
- ドキュメント内の要素が残したトラッキング項目の、詳細メッセージが表示される部分（下）

図 3.3. シナリオビルダ - ドキュメントログのタブ

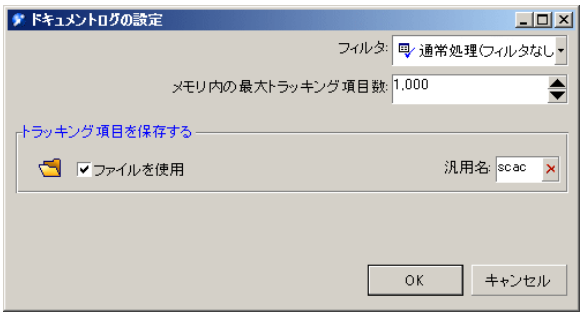


ドキュメントログの設定

ドキュメントログを設定するには以下の操作を実行します。

- ドキュメント処理のエラーの種類に基づいて、トラッキング項目にフィルタをかけます。
- ドキュメントログに表示されるトラッキング項目の最大数を指定します。
- ログに表示されるトラッキング項目を保存するかどうか決めます。

図 3.4. ドキュメントログの設定ウィンドウ



ドキュメントログを設定するには、［ログ／ドキュメントログを設定する...］メニューを選択します。

トラッキング項目にフィルタをかける場合は、［フィルタ］フィールドでオプションを選択します。

ドキュメントログに表示されるトラッキング項目の最大数を指定するには、［メモリ内の最大トラッキング項目数］フィールドに希望の項目数を入力します。

注意:

コンピュータのメモリ容量を考慮に入れずに表示されるトラッキング項目数を設定すると、処理時間が過大になる恐れがあります。

ドキュメント、ドキュメントのトラッキング項目と、シナリオの構成要素が残したトラッキング項目を保存するには、

- **［ファイルを使用］オプションをオンにします。**
- **［トラッキング項目を保存する］枠内の［汎用名］フィールド内に汎用名を入力します（例：acsc）。**

  汎用名は以下の内容を含む3つのファイルを作成します。
  - **シナリオに生成されるドキュメント：「.dat」ファイル（例：acsc.dat）**
  - **ドキュメントログ内のメッセージ（フィールドに関する警告、処理の失敗など）：「.msg」ファイル（例：acsc.msg）**
  - **「.dat」ファイルと「.msg」ファイル内のデータを迅速に検索するためのインデックスファイル：「.idx」ファイル（例：acsc.idx）**

注意:

［汎用名］フィールドにはファイル用の完全パスまたは相対パスを入力できます。例えば、

- c:\test\<汎用名>
- ..\<汎用名>

ディスク内の記憶とメモリを削除する

［ログ］メニューには2つの削除コマンドがあります。

- **［表示されたトラッキング項目を削除する］**

  このコマンドはシナリオビルダ内（Connect-Itログとドキュメントログ）のログ項目を削除します。トラッキング項目を含むファイルは、このコマンドでは削除されません。
- **［格納されたトラッキング項目を削除する］**

  トラッキング項目を保存している場合、シナリオの起動時に作成された3つのファイル（「.dat」、「.msg」、「.idx」）をこのコマンドで削除できます。

ドキュメントログを再び読み込む

［ログ／ドキュメントログを再び読み込む］コマンドにより、「.dat」、「.msg」と「.idx」ファイルにあるトラッキング項目全部を、ドキュメントログのタブ内に表示できます。このコマンドは、ドキュメントログ内のトラッキング項目の表示数を制限した場合に非常に有用です。この場合、このコマンドでドキュメント処理中に発生した問題点全てを表示できます。

トラッキング項目

ドキュメントログはトラッキング項目を表示します。各項目はシナリオの1構成要素に処理されたドキュメントに相当し、以下の内容を含んでいます。

- **識別番号**
- **ドキュメントが処理された方法を表すアイコン**
- **処理の日時**
- **処理されたドキュメントタイプ**
- **ドキュメントを生成した、または取り込んだ構成要素の名前**
- **ドキュメントの内容**

識別番号はドキュメントの処理順に付けられています。識別番号が1のドキュメントは一番最初に処理されています。ドキュメントがメモリに保存されている場合、固有の識別番号によりドキュメントが検索しやすくなっています。

ドキュメントログの各列を昇順または降順で並べ替えるには、各列のタイトルをクリックします。

上向きの青の三角形は、並べ替えが昇順であることを示しています。下向きの三角形は並べ替えが降順であることを示しています。列が並べ替えられていない場合、三角形は灰色です。

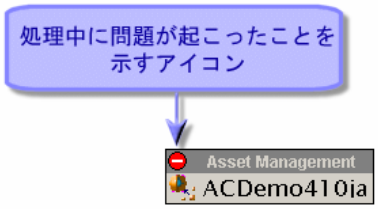
項目の一部が切れて表示されている場合、マウスポインタを項目上に置くと、項目全体がヒントの形で表示されます。

ドキュメントログ内で使用されるアイコン

アイコンは最も重要な情報に対応しています。例えば、同じドキュメント内のあるフィールドは警告を受けただけなのに対し、他のフィールドが拒否されたとすると、⛔アイコンのみが表示されます。これは、ドキュメントの完全な拒否は警告よりも重要だと見なさるためです。

シナリオ図内では、これらのアイコンがコンポーネントの開閉を表す電球のアイコンの代わりに表示されます。アイコンは、コンポーネントが少なくとも1つのドキュメントを正常に処理しなかったこと（1フィールドに対する警告が発生した、フィールドが拒否された、またはドキュメントが完全に拒否されたこと）を示しています。

図 3.5. 処理中に起こった問題



問題発生のアイコンがコンポーネント上に表示された場合、コンポーネント上を右クリックして［**トラッキング項目の表示**］をショートカットメニューから選択します。この時点で、ユーザは問題の発生を認識しているとみなされるため、アイコンは画面から消えます。

トラッキング項目のフィルタ

ドキュメントログ内に表示されるトラッキング項目には、次の条件に従ってフィルタをかけることができます。

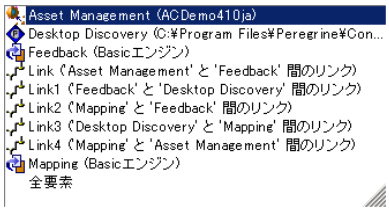
- ドキュメントを処理した構成要素
- ドキュメントの処理方法

シナリオ図内で選択されている構成要素がないと、メモリにあるトラッキング項目全体がドキュメントログに表示されます。

ある要素により処理されたドキュメントのトラッキング項目のみを表示するには、

- シナリオ図内でこの要素を選択します。または
- 【要素】フィールドで1項目を選択します。

図 3.6. ドキュメントログ内のトラッキング項目へのフィルタ

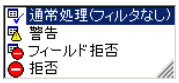


処理結果に応じてトラッキング項目を表示する場合、【フィルタ】フィールドで1項目を選択します。

選択した処理結果のトラッキング項目と、それよりも重要度の高い問題の項目が表示されます。

例：【フィールド拒否】を選択すると、フィールドを拒否されたドキュメントだけでなく、全体を拒否されたドキュメントも表示されます。

図 3.7. トラッキング項目のフィルタ



ドキュメントの詳細枠

ドキュメントの詳細枠では、ドキュメントのフィールド値についての情報と、ドキュメントがコンポーネントまたはリンクにより処理された方法を知ることができます。

ドキュメントログ内で1つのトラッキング項目が選択されると、そのドキュメントの詳細事項がログの右側にある枠内に表示されます。

ドキュメントはツリー構造で表示されます。

このツリー構造内の各ノードは以下の内容を表しています。

- ノードに対応する要素の名前
- ドキュメント内のノードの値
- ドキュメントを処理したコンポーネントまたはリンクが残すトラッキング項目

ドキュメントの詳細枠にある各トラッキング項目にはメッセージがあります。メッセージは、ドキュメント全体が拒否された理由などを説明します。

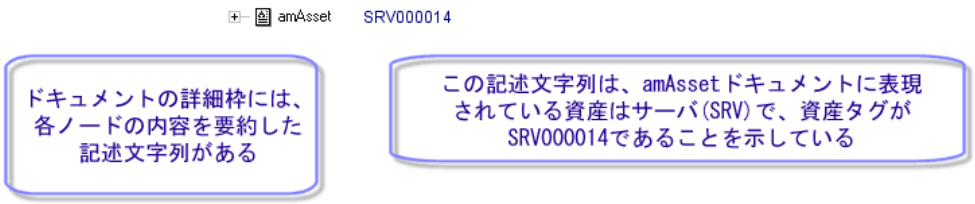
ドキュメントの詳細とドキュメントタイプの詳細

ドキュメントタイプの詳細は、生成用ドキュメントタイプ、または取り込み用ドキュメントタイプの編集時に表示されます。ドキュメントと同様に、ドキュメントタイプにはツリー構造があります。しかし、ドキュメントとドキュメントタイプの間には多くの相違点があります。

ドキュメントの詳細枠では、各ノードの値が表示されます。構造体またはコレクションの場合、この値は構造体またはコレクション内のフィールド値を要約した記述文字列です。例えば、AssetCenterの資産に対応するドキュメントのルートノードには、資産タグが記述（灰色で表示）されます。コレクションには、コレクション内の構成要素の数が表示されます。

ドキュメントタイプの詳細にはノードの名前だけが表示されます。

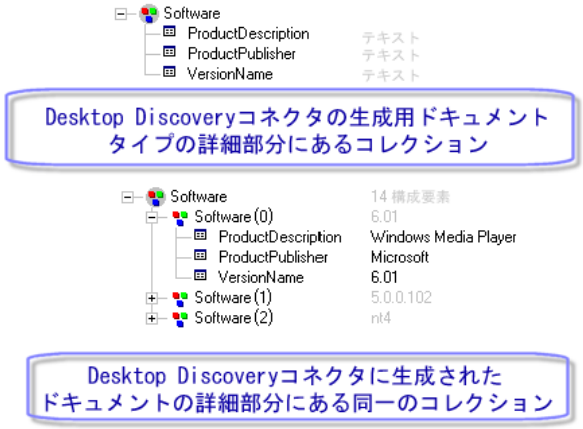
図 3.8. ドキュメント内の記述文字列



ドキュメントの詳細枠では、コレクション内の各構成要素はコレクションの子構造体として表示されます。これらの子ノードには、コレクションの名前と括弧で囲まれた数字が付きます。3構成要素のあるコレクションでは、第1構成要素には0、第2構成要素には1、第3構成要素には2が付きます。

ドキュメントタイプの詳細では、コレクションと、各フィールド値の性質（テキスト、整数、日付）のみが表示されます。

図 3.9. ドキュメントタイプ詳細内のコレクションと、ドキュメント詳細内のコレクション



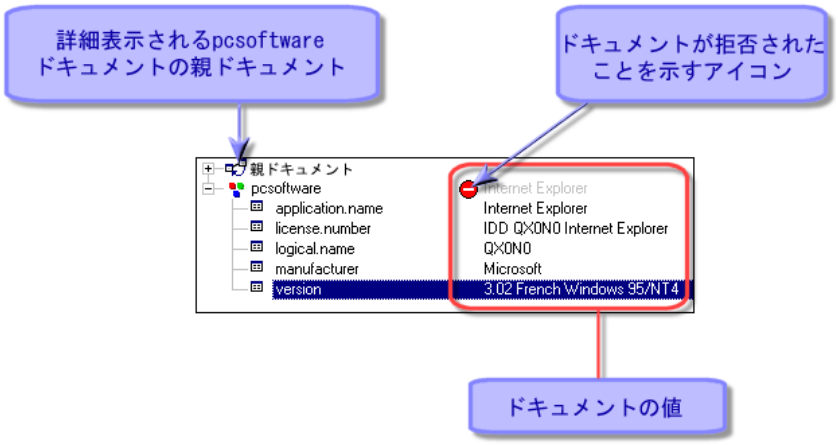
親ドキュメント

ドキュメントの詳細枠には「親ドキュメント」が含まれていることがあります。この親ドキュメントには、ドキュメント作成に使用されたデータが入っています。このため、マッピングボックスに生成されたドキュメントの詳細枠内の親ドキュメント下には、ドキュメント生成に使用されたソースドキュメントの詳細が含まれています。

例：「iddac.scn」シナリオでは、マッピングボックスに生成されたドキュメントの詳細枠内の親ドキュメントは、Desktop Discoveryコネクタに生成されたドキュメントの詳細を含んでいます。後者のドキュメントはマッピング内のソースドキュメントタイプにあたります。

マッピングボックスがドキュメントのフィールドを拒否した場合、親ドキュメントの詳細部分で、拒否されたフィールドの値と拒否の理由を確認することができます。

図 3.10. ServiceCenterコネクタに取り込まれた「pcsoftware」ドキュメントの詳細部分



ドキュメントの詳細に対応するXML文書（ドキュメント）を取り扱う

Connect-It内で生成されたドキュメントの詳細情報はXML文書です。以下の操作が可能です。

- このXML文書をコピーする
- このXML文書のDTD（文書型定義）をコピーする
- このXML文書を開く

ドキュメントの詳細に対応するXML文書をコピーする

1. ドキュメントの詳細枠内にポインタを置きます。
2. 右クリックします。
3. 表示されるショートカットメニューから、［このXML文書をコピーする］を選択します。

ドキュメントの詳細のDTDをコピーする

1. ドキュメントの詳細枠内にポインタを置きます。
2. 右クリックします。

3 表示されるショートカットメニューから、［このXMLのDTDをコピーする］を選択します。

ドキュメントの詳細に対応するXML文書を開く

1 ドキュメントの詳細枠内にポインタを置きます。
2 右クリックします。
3 表示されるショートカットメニューから、［このXML文書を開く］を選択します。

注意:

XML文書は、Connect-Itをインストールしたコンピュータで、「.xml」ファイルに関連付けられたアプリケーション上で開かれます。例えばInternet ExplorerやNetscapeなどです。

ドキュメントの詳細内で要素を検索する

検索機能を使用すると、ドキュメントログのドキュメントの詳細内で要素を検索できるようになります。

ドキュメントの詳細内で要素を検索する

1 ドキュメントの詳細で要素を1つ選択します。
2 右クリックします。
3 ［検索］ショートカットメニューを選択します。
4 ダイアログボックスが表示されます。
5 ［検索する文字列］フィールドに、検索する要素の名前を入力し、以下のオプションを必要に応じて選択します。
   - 大文字と小文字を区別
   - 検索方向（［上へ］または［下へ］）
6 ［検索］をクリックします。

次の要素または前の要素を検索するには、

1 以下の操作の内1つを実行します。
   - 右クリックして、ショートカットメニューから［次を検索］（「F3」キー）を選択します。
   - 右クリックして、ショートカットメニューから［前を検索］（「Shift＋F3」キー）を選択します。

注意:

検索の範囲は、ドキュメントの詳細に表示されている要素に制限されます。サブノード内も検索する場合は、検索の前にサブノードを開いておく必要があります。

ドキュメントの詳細枠内で使用されるトラッキング項目

トラッキング項目は、シナリオのコンポーネントまたはリンクがドキュメントのフィールド上に実行した処理に関する情報です。

ドキュメントの詳細枠内では、トラッキング項目にアイコンが付きます。各トラッキング項目の内容は、別の枠に表示されるメッセージで説明されます。

トラッキング項目は2つに分類されます。

- 情報を提供するトラッキング項目（フィールドに関する重要な情報、または詳細情報）
- 処理中に発生した問題に関するトラッキング項目（フィールドに対する警告、処理の失敗）

処理エラーのトラッキング項目 のみが、ドキュメントの完全な拒否または部分的な拒否を意味します。

トラッキング項目、アイコンの意味と、項目を説明するメッセージのタイプは、以下の表の通りです。

トラッキング項目	意味	メッセージのタイプ	例
	詳細情報	データ処理に関する技術的な詳細情報	「INDコネクタはドキュメントに転換するデータをダウンロードします。」ファイルの大きさ（KB）やダウンロードの速度などの情報
	重要な情報	データ使用に重要な情報	ドキュメントの取り込み後にコネクタが作成したフォルダとファイルの名前

トラッキング項目	意味	メッセージのタイプ	例
⚠	フィールドに対する警告	フィールドが適切に処理されなかった理由	「ServiceCenterコネクタはフィールドを取り込みましたが、ServiceCenterデータベースの構造に従ってフィールドの一部を切り詰めました。」
⛔	処理の失敗	処理の失敗の理由	「予期されていないフィールド値があったためフィールドが拒否されました。ドキュメントの部分的拒否または完全な拒否が発生する可能性があります。」

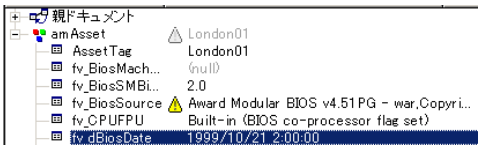
ノードの横に現れる灰色のトラッキング項目

ドキュメントの詳細枠では、トラッキング項目が親ノードの横に灰色で表示されることがあります。

灰色のトラッキング項目は、ノードの子ノードの内少なくとも1つに、トラッキング項目があることを意味します。

ノード内に複数のトラッキング項目がある場合、最も重要だと見なされるトラッキング項目が、親ノードの横に灰色で表示されます。

図3.11. ノード下にトラッキング項目があることを意味する灰色のトラッキング項目



トラッキング項目のタイプによるドキュメントのフィルタ

トラッキング項目のタイプに基づいてドキュメントの詳細にフィルタをかけると、選択したトラッキング項目タイプを含むノードのみが表示されます。

拒否されたノードのみを表示する場合は、ドキュメントの詳細枠の上にある［**トラッキング項目のフィルタ**］フィールドで ⛔ を選択します。

このドロップダウンリストから1項目を選択すると、選択項目より下にあるトラッキング項目アイコンを含むノードも全て表示されます。

ドロップダウンリストでは、情報の重要度の低い項目から高い項目順に並べられています。つまり重要度の低い［詳細モード］が1番上にあり、重要度の高い［エラー］（処理の失敗）が1番下にあります。

図 3.12. ドキュメント詳細内のトラッキング項目用フィルタ

トラッキング項目の説明メッセージ

ドキュメントの詳細枠内のトラッキング項目は、メッセージで説明されます。

ドキュメントの詳細枠内のトラッキング項目を1つ選択すると、項目の内容を説明するメッセージが［メッセージ］枠内に現れます。この枠はドキュメントログの下部にあります。

例：Asset Managementデータベースの保全性を破壊する値を含むフィールドが、Asset Managementコネクタにより拒否されたことを説明するメッセージ。

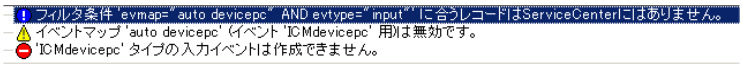
選択したノードの子ノードのメッセージを全て表示する

ドキュメント詳細の親ノード下の子ノードに含まれている全メッセージを表示するには、  をクリックします。

例

 オプションが選択されていないと、選択したノードのメッセージのみが表示されます。

が選択されていると、子ノードの全メッセージが表示されます。



オプション

シナリオビルダのオプションには以下のカテゴリがあります。

- **アクセスビリティ**
- **表示**
- **確認**
- **コネクタ**
- **文書**
- **ログ**

シナリオビルダのオプション用ウィンドウを開くには、**［編集／オプション...］**メニューを選択します。

このウィンドウでは、**［値］**列からセルを選択してダブルクリックし、オプションを変更します。表示されるテキストゾーンに値を入力します。

注意:

デフォルトのオプション値を変更すると、変更した値の名前は赤色になります。これにより、値を変更したオプションを素早く検索できます。

アクセスビリティ

この項目には、シナリオビルダの使用を快適にするためのオプションがまとめられています。

- 背景の画像を削除する
- フラッシュを避ける
- 色を避ける
- フォントを太字にする

注意：これらのオプションはシナリオビルダでは使用されていません。ペレグリンシステムズの他のアプリケーションで使用されています。

表示

この項目には、シナリオビルダの表示オプションがまとめられています。

認証証明書に認証されたコネクタのみ表示する

このオプションにより、シナリオビルダのツールボックスには、認証証明書に使用を許可されたコネクタのみが表示されます。

ウィザード内の説明の色を変更する

このオプションではウィザード内の説明の色を変更できます。例えば設定ウィザード内の説明の文字の色などです。

フィールドで［Esc］キーを無効にする

Windows上では、［Esc］キーを使用すると編集ウィンドウが閉じるようになっています。例：ソースコネクタの生成用ドキュメントタイプの編集ウィンドウ。

シナリオビルダ内では、安全のために［Esc］キーは現在編集している要素のみに適用されます（このオプションは［いいえ］に設定されています）。例：マッピングの編集ウィンドウでのコレクションからコレクションへのマッピングの編集。

［Esc］キーの動作をWindowsの標準的動作に戻すには、このオプションを選択してください。

Windowsグラフィックス

このオプションでは、シナリオビルダの表示をフラットまたは標準にできます。

詳細モード

このオプションを使用するとビルダは詳細モードで起動します。アプリケーションを、コマンドプロンプトを使って「-verbose」オプションで起動すると、このオプションの値は無視されます。

プレビュー時にブロック内に取得するドキュメントの数

このオプションでは、ドキュメントのプレビューウィンドウで表示するドキュメントの数を指定できます。デフォルトのドキュメント数は20です。

ドキュメントの生成テスト中に生成するドキュメントの数

このオプションでは、ドキュメントの生成テスト中に生成するドキュメントの数を指定できます。テストモードでは、シナリオのソースコネクタはドキュメントを生成しますが、ターゲットコネクタは外部アプリケーションとは相互作用しません。ドキュメント数を制限するには、チェックボックスをオンにする必要があります。

このオプションは、生成用ドキュメントタイプをテストする際に自動的に使用されます。しかし、シナリオがWindowsのサービスとして実行されると、このオプションは自動的に無効になります。

Connect-Itログ内に表示するトラッキング項目の最大数

このオプションにより、Connect-Itログに表示されるトラッキング項目の最大数を制限できます。

1回に表示する子ノードの最大数

このオプションにより、ドキュメントタイプとドキュメント内の親ノード下に表示される子ノードの数を指定できます。例：ルートノード下ではドキュメントタイプの最初の3コレクションのみを表示する。

1ノードにつき自動的に展開する子ノードの最大数

このオプションにより、ドキュメントタイプとドキュメント内の親ノード下に表示される子ノードの数を指定できます。ドキュメントタイプのノードに多数の子ノードがある場合、このオプションは非常に有用です。

自動的にマップするレベルの最大数

このオプションでは、テーブルからテーブルへのマッピングまたはノードからノードへのマッピング時に、自動的にマップされるノード（構造体またはコレクション）のレベル数を指定できます。

起動時に［ご存知でしたか］ウィンドウを表示する

このオプションを選択すると、Connect-Itの起動時に［ご存知でしたか］ウィンドウが表示されます。このオプションは［ご存知でしたか］ウィンドウで直接オフにすることもできます（**スタートアップ時に表示する**）。

タブ

Connect-Itのタブを以下のオプションを用いて変更できます。

- **イメージの表示**
  このオプションでは、タブ名の横にあるアイコンの表示／非表示を選択します。例：［ドキュメントログ］タブの  アイコン
- **ヒントの表示**
  このオプションでは、タブの上にポインタを置くと現れるヒントの表示／非表示を選択します。
- **テキストの表示**
  このオプションでは、タブのタイトルの表示／非表示を選択します。
- **タブの表示スタイル**
  このオプションでは、生成用ドキュメントタイプ用ウィンドウのタブの表示スタイルを変更できます。

ヒントを表示する

このオプションを使用すると、Connect-It内でメニューやコンポーネントなどの上にポインタを置くとヒントが表示されます。

確認

一定の操作の後に、プログラムがダイアログボックスを通じてユーザに確認を取るように設定するオプションが、このグループにまとめられています。

コレクションからコレクションへのマッピングが必要な場合に、警告を表示する

このオプションにより、コレクションからコレクションへのマッピングが必要な場合に、警告が表示されます。例：ソースコレクションのフィールドがターゲットコレクションのフィールドにマップされても、2つのコレクションがお互いにマップされていない場合、警告が表示され、2つのコレクションのマッピングが必要であることをユーザに知らせます。

一時停止モードの起動時に、メッセージを表示する

このオプションにより、一時停止モードの起動毎にメッセージが表示されます。

JVMパラメータが変更されるたびにメッセージを表示する

このオプションにより、JVMパラメータが変更されるたびにメッセージが表示されます。

コンパイルのエラーを含むスクリプトの認証を許可しない

このオプションを選択すると、シンタックスエラーを含むスクリプトの認証は許可されません。

1コネクタの始動時に現在のシナリオの全コネクタを開く

このオプションにより、1コネクタの起動時に現在のシナリオの全コネクタを開くことができます。このオプションにより、コネクタを1つずつ開くことなくシナリオのマッピングを素早く起動することが可能になり、非常に便利です。

編集するドキュメントタイプを選択するために、ダイアログボックスを使用する

このオプションにより、開かれたマッピングボックスやコネクタをダブルクリックすると、ダイアログボックスが表示され、ドキュメントタイプやマッピングを選択できるようになります。

コネクタ

以下のコネクタ用のオプションがあります。

- LDAPコネクタ
- InfraTools Network Discoveryコネクタ
- ServiceCenterコネクタ
- テキストコネクタとXMLコネクタ

その他に、全コネクタ用のオプションがあります。

- **トラッキング項目内にクエリを表示する**
- **データベース**

  フィールドのデータ型がサポートされていない場合メッセージを表示する
- **処理するドキュメント数を計算する**
- **サーバとの時間差に応じて日時を調整する**
- **設定ウィザードのキャッシュ使用のオプションをデフォルトで有効にする**

LDAPコネクタ

［ルートDSEで定義される属性を表示する］オプションにより、サーバの現在の日付の取得を可能にする属性を、LDAPサーバが表示するかどうかを指定できます。

InfraTools Network Discoveryコネクタ

［起動日以前のイベントのみを処理する］オプションにより、シナリオを起動すると、起動前に起こったイベント全てを処理できます。イベントの数が多いと、シナリオが停止する恐れがあります。デフォルトではこのオプションは**［いいえ］**になっています。

ServiceCenterコネクタ

［ServiceCenterコードページ番号を強制する］オプションでは、自分で選択したコードページ番号を選択できます。デフォルトではこのオプション値は「0」です。この値は、ServiceCenterのコードページが、Connect-Itのインストールされたコンピュータのコードページであることを意味しています。

［フィルタのデータ型を確認する］オプションでは、ServiceCenterのフィールド値タイプが、クエリ作成者に指定されたタイプに一致するかどうかを調べることができます。例えば、**日付**型のフィールドが、整数に比較されている「sysmodtime > 3」フィルタは無効です。この場合エラーメッセージが表示され、コネクタはドキュメントを生成しません。

［コネクタが使用可能なドキュメントタイプを発行する間、進行状況を表示する］オプションを使うと、Connect-Itログ内に、開いたコネクタが発行する使用可能なドキュメントタイプの名前を表示することができます。

データベース

［フィールドのデータ型がサポートされていない場合メッセージを表示する］オプションを使用すると、フィールドがコネクタによってサポートされていない場合、ドキュメントログ内にメッセージが表示されます。

XMLと区切られたテキスト

［Connect-Itログ内に処理中のURLを表示する］オプションを選択すると、XMLコネクタと区切られたテキストコネクタが処理しているURLを、Connect-Itログ内に表示することができます。

トラッキング項目内にクエリを表示する

このオプションにより、ドキュメントログ内で（メッセージ枠内）クエリのテキストを表示できます。これはシナリオのテスト段階やデバッグ中に便利です。

処理するドキュメント数を計算する

このオプションにより、シナリオビルダのステータスバーに、処理されるドキュメント全体数の内いくつのドキュメントが処理されたかが表示されます。例えば、200の内4個のドキュメントが処理された場合、「4/200」がステータスバーに現れます。

警告:

処理するドキュメント数が多大な場合にこのオプションを使用すると、コンピュータの性能が低下する可能性があります。

サーバとの時間差に応じて日時を調整する

このオプションがオンになっていると、コネクタの設定時にサーバとの時間差を指定するたびに、読み取られたり書き込まれたりする日時は時間差の値に応じて調整されます。

注意：時間差の調整はConnect-It 2.7.1までは自動的に実行されていました。

サーバとの時間差のオプションについては、マニュアル『**コネクタ**』の「**コネクタの設定**」の章の「**サーバとの時間差を指定する**」の節を参照してください。

設定ウィザードのキャッシュ使用のオプションをデフォルトで有効にする

このオプションにより、コネクタでキャッシュファイルを使用するために、コネクタの設定ウィザードでキャッシュの使用を手動で選択する必要がなくなります。

キャッシュオプションについては、マニュアル『**コネクタ**』の「**コネクタの設定**」の章の「**キャッシュを設定する**」の節を参照してください。

文書

注意：この項目の文書は、コネクタに処理されるドキュメントではなく、シナリオを含むSCNファイルのことを指しています。

この項目は、SCNファイルの読み込みに関するオプションをまとめています。

- **［ファイル］メニューに格納する、最近開いた文書の最大数**

  このオプションは、**［ファイル］**メニューのリストに表示されるSCNファイルに関するものです。
- **起動時に前回最後に使用した文書を自動的に読み込む**
- **保存前に確認を要求する**

このオプションにより、Connect-Itで保存を実行する前にダイアログボックスが表示されます（マッピング、シナリオ、スケジューラなどの保存）。

ログ

ログ用には［**ログファイルのサイズ（MB）**］オプションがあり、「.log」ファイルの大きさを指定できます。サイズの最大値に達するたびに、新規データは最も古いデータを削除して上書き保存されます。

4 統合シナリオのインプリメンテーション

統合シナリオのインプリメンテーションでは以下の手順に従います。
- コンポーネントを選択する。
- コンポーネント同士をリンクする。

ウィザードでシナリオをインプリメントする

シナリオインプリメンテーションウィザードでは以下の操作を実行できます。
- シナリオのソースコネクタとターゲットコネクタを選択する。
- 2コネクタの設定を実行する。

ターゲットコネクタの設定が終了すると、2つのコネクタはマッピングボックスにリンクされます。コネクタとマッピングボックスは、シナリオビルダのシナリオ図枠内に表示されます。

シナリオインプリメンテーションウィザードを起動する

1 以下のアクションのいずれかを実行します。
- シナリオビルダのツールバーで をクリックします。または、

- **［ファイル／新規作成］**メニューを選択します。

手動でシナリオをインプリメントする

シナリオを手動で作成するには、

1 シナリオ図内にコンポーネントを配置します。

2 コンポーネントをリンクします。

シナリオ図内にコンポーネントを配置する

コネクタをシナリオ図内に置くと、自動的に**［コネクタの設定］**ウィザードが起動します。

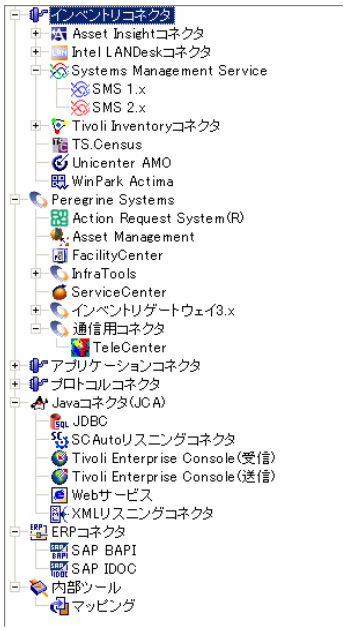
シナリオ図内にコンポーネントを配置する

1 ツールボックスからシナリオ図へコンポーネントをドラッグします。またはコンポーネントを選択しダブルクリックします。

2 表示される**［コネクタの設定］**ウィザードページに値を入力します。

コネクタの設定については、マニュアル**『コネクタ』**の**「コネクタの設定」**の章を参照してください。

図 4.1. シナリオビルダのツールボックス



コンポーネントを設定し直す

1. シナリオ図内でコンポーネントを選択します。
2. 以下のアクションのいずれかを実行します。
   - **［ツール／設定］**メニューを選択します。
   - 右クリックし、表示されるショートカットメニューから**［コネクタを設定する...］**を選択します。

コンポーネントをリンクする

コネクタをリンクする方法により、どのコネクタがソースコネクタまたはターゲットコネクタになるかを定義できます。

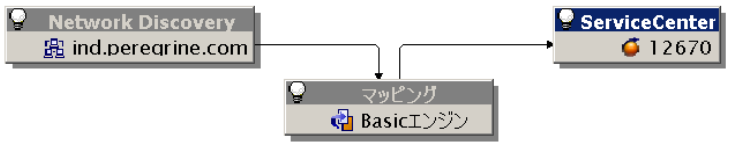
- コネクタが外にリンクを送信する場合、このコネクタはソースコネクタになります。
- コネクタがリンクを受信する場合、このコネクタはターゲットコネクタです。

注意:

ピボットドキュメントタイプを使用しない場合、ソースコネクタをターゲットコネクタに直接リンクすることは稀です。原則としてソースコネクタをマッピングボックスに連結し、マッピングボックスをターゲットコネクタに連結します。

以下の例では、Network Discoveryコネクタはソースコネクタで、ServiceCenterコネクタはターゲットコネクタです。Network Discoveryコネクタがドキュメントを生成すると、ドキュメントはまずマッピングボックスで変換され、その後ServiceCenterコネクタに取り込まれます。

図 4.2. シナリオのコンポーネントのリンク



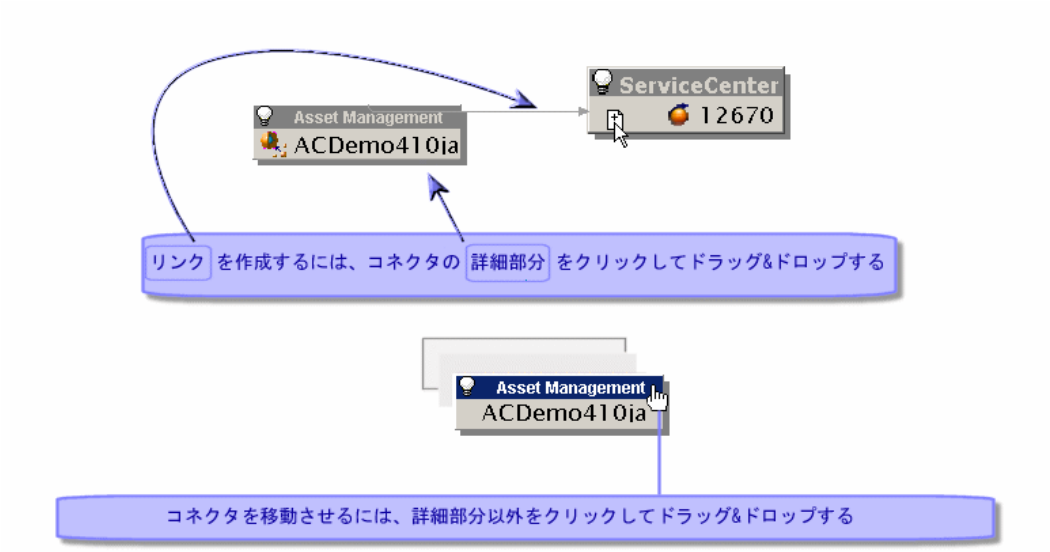
コンポーネントを別のコンポーネントにリンクする

1. コンポーネントの下の部分をクリックします。
2. コンポーネントの詳細部分が白で表示されます。
3. マウスの左ボタンを押したままにします。
4. 表示されるリンクを別のコンポーネントへドラッグします。

注意:

コンポーネントの詳細部分以外をクリックしてドラッグすると、コンポーネントをシナリオ図内で移動させることになります。

図 4.3. リンクの作成またはコネクタの移動



マッピングボックスを通さずに2コネクタをリンクする

2コネクタをリンクすると、Connect-Itは自動的に2コネクタ間にマッピングボックスを作成します。マッピングボックスを作成しない場合は、［Shift］キーを押したまま2コネクタをリンクします。

2コネクタ間の直接リンクを使用するのは、一部のコネクタ（テキストコネクタやXMLコネクタ）のみです。直接リンクにより、ソースコネクタに生成されるドキュメントの構造を利用して、ソースコネクタのデータを直接エキスポートすることができます。

コンポーネントを削除する

シナリオ図内のコネクタ、リンクまたはマッピングボックスを削除するには、

1. コンポーネントを選択します。
2. 以下の操作の内1つを実行します。
   - ［編集／削除］メニューを選択します。
   - キーボードの［Del］キーを押します。
   - ショートカットメニューから［削除］を選択します。

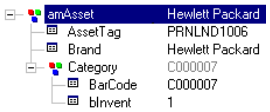
生成用または取り込み用ドキュメントタイプを編集する

生成用または取り込み用ドキュメントタイプはテンプレートの役目を果たしており、コネクタはシナリオの実行時に、このテンプレート使用してドキュメントを生成したり取り込んだりします。生成用または取り込み用ドキュメントタイプは、コネクタが発行する使用可能なドキュメントタイプから選択されたフィールドの集まりです。

例：

idd \iddac41 \iddac.scnシナリオには、Asset Managementコネクタの取り込み用ドキュメントタイプ「**amAsset**」があります。この取り込み用ドキュメントタイプは、Asset Managementコネクタが発行する使用可能なドキュメントタイプ「**amAsset**」内の全フィールドから選択された一部のフィールド群です。

Connect-Itで処理されるドキュメントタイプとドキュメントの関係は、次の表の通りです。

使用可能なドキュメントタイプ	生成用または取り込み用ドキュメントタイプ	生成されるドキュメント
		
設定 コネクタは全部の使用可能なドキュメントタイプを発行します。ユーザはこの内1つの使用可能なドキュメントタイプを選択します。	**ドキュメントタイプの作成** 生成用、または取り込み用ドキュメントタイプを作成するには、使用可能なドキュメントタイプから必要な要素を選択します。	**シナリオの起動** コネクタは、生成用または取り込み用ドキュメントタイプに沿って、ドキュメントの生成または取り込みを実行します。

コネクタの生成用ドキュメントタイプ、または取り込み用ドキュメントタイプを作成するには、

- コネクタが発行する使用可能なドキュメントタイプから、1つのドキュメントタイプを選択します。

- 選択したドキュメントタイプから必要なフィールドを選択します。

例：

Asset Managementコネクタが発行する全部の使用可能なドキュメントタイプから、「amProduct」ドキュメントタイプを選択します。次に、必要なフィールド（バーコード、メーカー、価格、カタログ番号など）を選択します。

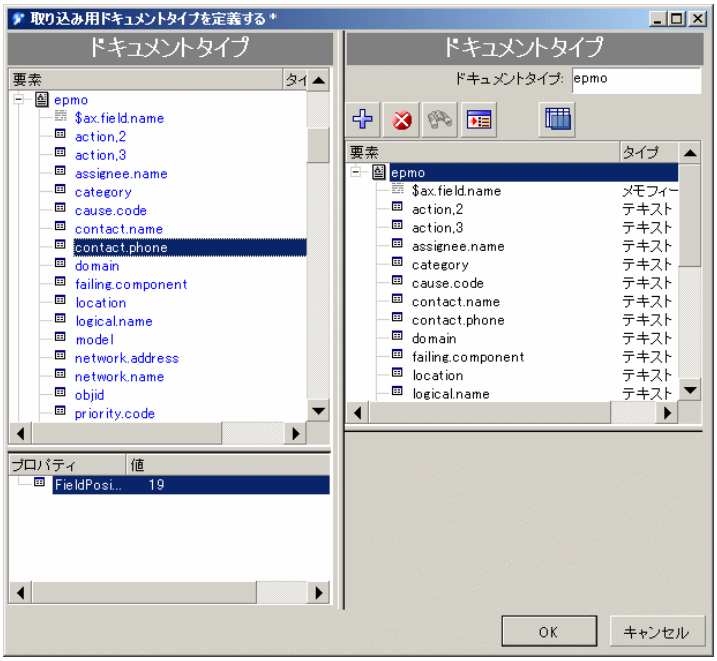
生成用または取り込み用ドキュメントタイプの編集ウィンドウにアクセスする

生成用または取り込み用ドキュメントタイプの編集ウィンドウは、2つのゾーンから成っています。

- 生成用または取り込み用ドキュメントタイプを作成するための、画面右側の作業枠

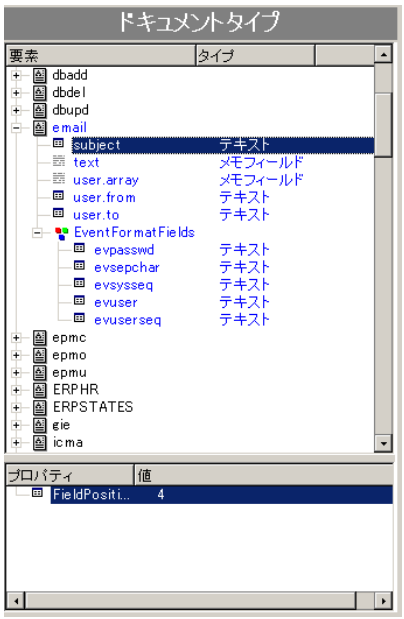
  この作業枠の下にはルール（ディレクティブ）用のタブが表示されます。ディレクティブのないコネクタもあります。
- コネクタが発行する使用可能なドキュメントタイプが表示される画面左側のゾーン。このゾーンは2つの部分に分かれています。
  - 使用可能なドキュメントタイプのルートノード（🖻）が表示される枠。この枠内でドキュメントタイプのノード（ルートノード、構造体、コレクション）内を全部表示すると、ノード内の全要素を参照できます。各ノードと各フィールド（端末ノード）には要素の名前とタイプ（テキスト、整数、日付...）が表示されます。ドキュメントタイプ内で1ノードを選択すると、このノードは現在の選択項目になります。
  - 現在の選択項目のプロパティと値を表示する枠。これらの技術的情報は、選択項目の意味やプロパティを理解するのに利用されます。

図 4.4. ServiceCenterコネクタの取り込み用ドキュメントタイプの編集



以下の画面では、「email」ドキュメントタイプの「subject」フィールドのタイプは「テキスト」です。

図 4.5. 使用可能なドキュメントタイプのゾーン



生成用または取り込み用ドキュメントタイプの編集ウィンドウにアクセスする

1. シナリオ図部分でコネクタを選択します。
2. **［ドキュメントタイプ］**タブを選択します。またはコネクタをダブルクリックします。
3. 生成用ドキュメントタイプ、または取り込み用ドキュメントタイプの枠を選択します。
4. をクリックすると、ドキュメントタイプを新規に作成できます。 をクリックすると、選択した既存のドキュメントタイプを編集できます。

生成用または取り込み用ドキュメントタイプを作成する

生成用または取り込み用ドキュメントタイプを作成するには、ルートノードまたは要素（コレクション、構造体、フィールド）を、ドキュメントタイプの編集ウィンドウの作業枠内に移動させます。

生成用または取り込み用ドキュメントタイプを作成する

生成用または取り込み用ドキュメントタイプを作成するには、

1 ルートノードまたはドキュメントタイプの1つまたは複数の要素を、以下の方法で作業枠内に置きます。

- 選択した要素を、作業枠にドラッグ&ドロップします。
- 必要な要素を選択し、＋をクリックします。
- 要素またはルートノードを直接ダブルクリックします。

注意:

ノード（ルートノード、構造体、コレクション）下にある多数のフィールドを使用する場合は、ノードを直接右の作業枠に移動させる方法を取ると便利です。ノード下の全フィールドも自動的に作業枠へ移るので、ここから不必要なフィールドを削除してドキュメントタイプを作成します。

作業枠から要素を削除する

1 要素を選択し ⊗ をクリックするかまたは［Del］キーを押します。

2 要素を画面左側の使用可能なドキュメントタイプ枠へドラッグして戻します。

要素の選択が終了したら、ドキュメントタイプに名前を付けます。（デフォルトでは、画面左から右の作業枠に移した使用可能なドキュメントタイプのノードの名前が付いています。）［OK］をクリックします。

注意:

マッピングボックスでマッピングを作成すると、マッピング内で使用されたソースコネクタの使用可能なドキュメントタイプの要素は、生成用ドキュメントタイプに自動的に追加されます。ドキュメントタイプの編集は以下の場合に便利です。

- マッピングで使用されなくなった要素をドキュメントタイプから削除する場合
- マッピングで使用されていない要素の内、必要な要素を追加する場合。例えばシナリオの実行時に、InfraTools Network Discoveryコネクタのスケジュールのポインタを更新するために必要なソース内の要素を追加する場合などです。

生成用または取り込み用ドキュメントタイプのDTDをコピーする

Connect-It内の各ドキュメントタイプは、それぞれXML形式のドキュメントに対応しています。Connect-Itでは、このドキュメントに関連したDTDをコピーできます。

生成用または取り込み用ドキュメントタイプのDTDをコピーする方法

1. 生成用ドキュメントタイプを作成または編集します。
2. ポインタを作業枠（画面右）に置きます。
3. 右クリックします。
4. **［このXMLのDTDをコピーする］**をショートカットメニューから選択します。

ドキュメントタイプの編集ウィンドウが開いている場合、右クリックするとショートカットメニューが表示されます。このメニューで、DTDをクリップボードへコピーできます。

生成用ドキュメントタイプのデータを参照する

Connect-Itでは、生成用ドキュメントタイプの要素に対応するデータを表示できます。表示されるのは、コネクタの接続先の外部アプリケーション内に記録されているデータです。

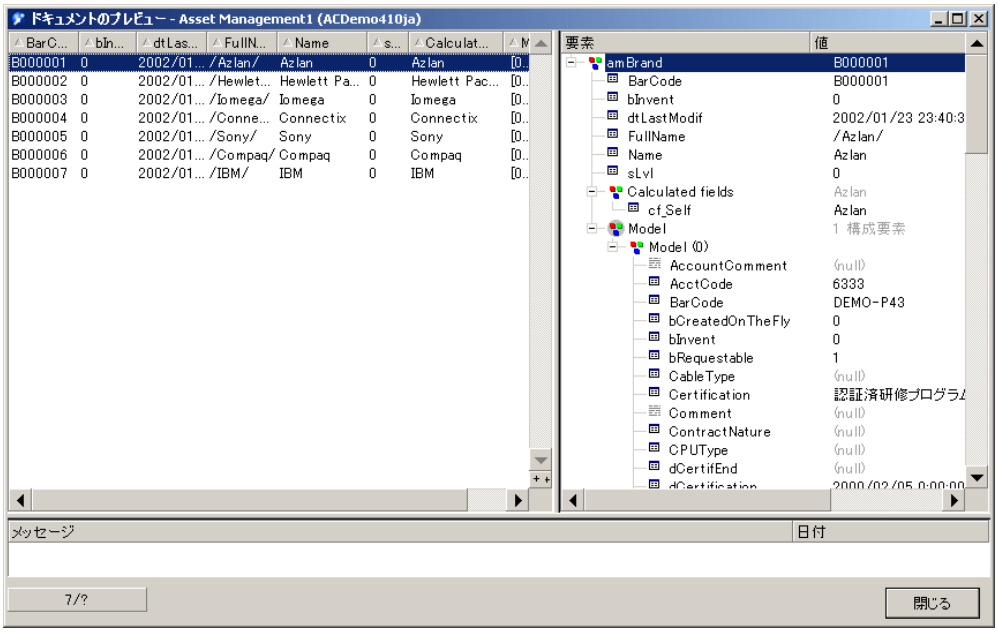
注意:

データの表示に一定以上の時間がかかると、進行中のアクションを中断するためのウィンドウが自動的に開きます。表示の中断を選択すると、既に検索されたデータにアクセスできるようになります。 をクリックすると、中断したアクションを再開できます。

例 :

データベースコネクタの生成用ドキュメントタイプのデータは、データベースの1つのテーブルにある複数のレコードに一致します。

図 4.6. 生成用ドキュメントタイプのデータプレビュー用ウィンドウ



プレビュー用ウィンドウ内に読み込まれるドキュメントの数は、［編集／オプション］メニューの表示項目内の［プレビュー時にブロック内に取得するドキュメントの数］オプションで指定されます。デフォルトでは、最初の20のドキュメントが取得されます。次のドキュメントを読み込むには、＋ ボタンを押します。全ドキュメントが表示されると、＋ ボタンは灰色になります。

注意:

＋ ボタンは一部のコネクタでのみ使用可能です。

生成用ドキュメントタイプを表示する

1. **生成用ドキュメントタイプを作成または編集します。**
2. **以下の操作の内1つを実行します。**
   - **をクリックします。**
   - **ポインタを作業枠上に置いて右クリックし、ショートカットメニューから［データを表示する...］を選択します。**

ドキュメントタイプのデータのプレビュー用ウィンドウは、2つの部分から成っています。

- 画面左側の枠には、ドキュメントタイプの要素に対応する外部アプリケーションの全レコードが表示されます。
  画面左の各列は、生成用ドキュメントタイプの各要素に対応します。
- 画面右側の枠には、生成用ドキュメントタイプと、左側の枠で選択したレコードの値が表示されます。

データの読み込みを中断する

1 以下の操作の内1つを実行します。
- 現在の操作を中断するためのダイアログボックスが表示されたら、［OK］をクリックします。
- キーボードの［Esc］キーを押します。

ドキュメントタイプ内の要素を検索する

検索機能を使用すると、ドキュメントタイプ内の要素を検索する際に、ドキュメントタイプの全ノードを開く必要がなくなります。

ドキュメントタイプ内の要素を検索する

1 作業枠内または使用可能なドキュメントタイプのゾーンで、ドキュメントタイプの1要素を選択します。
2 以下の操作の内1つを実行します。
- 右クリックして、ショートカットメニューから［検索（Ctrl + F）］を選択します。
-  をクリックします。

3 ［検索する文字列］フィールドに、検索する要素の名前を入力し、以下のオプションを必要に応じて選択します。
- 大文字と小文字を区別
- 検索方向（［上へ］または［下へ］）

4 ［検索］をクリックします。

次の要素または前の要素を検索するには、

1 以下の操作の内1つを実行します。
- 右クリックして、ショートカットメニューから［次を検索］（「F3」キー）を選択します。
- 右クリックして、ショートカットメニューから［前を検索］（「Ctrl+F3」キー）を選択します。

注意:

検索の範囲は、使用可能なドキュメントのウィンドウに表示されている要素に制限されます。サブノード内も検索する場合は、検索の前にサブノードを開いておく必要があります。

ドキュメントタイプ編集ウィンドウでのショートカットメニュー

ドキュメントタイプ編集ウィンドウには、2種類のショートカットメニューがあります。

- 使用可能なドキュメントタイプのゾーン（画面左側）にポインタが位置する時のメニュー
- 作業枠にポインタが位置する時

使用可能なドキュメントタイプのゾーン（画面左側）にポインタが位置する時のメニュー

コマンド	機能
検索	文字列検索用のダイアログボックスを表示します。
次を検索	ダイアログボックスに最後に入力された文字列を、ウィンドウの下方向に検索します。
前を検索	ダイアログボックスに最後に入力された文字列を、ウィンドウの上方向に検索します。
このXMLのDTDをコピーする	コンピュータのクリップボードに、トラッキング項目に対応するXMLのDTD（文書型定義）をコピーします。
パスをコピーする	コンピュータのクリップボードに、選択した要素のパスをコピーします。
この要素を追加する	作成中の生成用または取り込み用ドキュメントタイプに、選択した要素を追加します。
使用されていないドキュメントタイプにフィルタを適用する	作業枠内には、編集される生成用または取り込み用ドキュメントタイプのみが表示されます。

作業枠にポインタが位置する時

コマンド	機能
検索	文字列検索用のダイアログボックスを表示します。
次を検索	ダイアログボックスに最後に入力された文字列を、ウィンドウの下方向に検索します。
前を検索	ダイアログボックスに最後に入力された文字列を、ウィンドウの上方向に検索します。
このXMLのDTDをコピーする	コンピュータのクリップボードに、トラッキング項目に対応するXMLのDTD（文書型定義）をコピーします。
パスをコピーする	コンピュータのクリップボードに、選択した要素のパスをコピーします。
この要素を削除する	作成中の生成用または取り込み用ドキュメントタイプから、選択した要素を削除します。
データを表示する	生成用ドキュメントタイプの要素に対応するデータを表示します。
使用されていないドキュメントタイプにフィルタを適用する	作業枠内には、編集される生成用または取り込み用ドキュメントタイプのみが表示されます。

シナリオを保存する

シナリオの保存

1 以下の操作のいずれかを実行します。
- **［ファイル／保存］**を選択します。
- をクリックします。

別の名前を付けてシナリオを保存する場合
1 **［ファイル／名前を付けて保存］**を選択します。

シナリオのバックアップコピー

シナリオを保存すると、シナリオのフォルダ内にバックアップコピーが自動的に作成されます。このコピーの拡張子は「.bak」です。

このオプションを無効にするには、

1 **［編集／オプション］**を選択します。
2 **［確認］**ノードを開きます。
3 **［シナリオのバックアップコピーを保存する］**オプションの値を**「いいえ」**にします。
4 **［OK］**をクリックします。

シナリオの移動

シナリオをあるフォルダから別のフォルダへ移動させる場合、以下の操作のいずれかを必ず実行してください。

- シナリオの関連ファイルを移動させる
- シナリオスクリプト内の相対パスを変更する
  特に「.bas」、**「.str」**と**「.mpt」**ファイルのパスを確認してください。

既存シナリオを開く

シナリオビルダでは、拡張子「.scn」のファイルを開くことができます。

既存のファイルを開く

1 以下の操作のいずれかを実行します。
   - シナリオビルダを起動し、**［ファイル／開く］**を選択します。
   - エクスプローラで「.scn」ファイルを選択し、ダブルクリックします。
     シナリオビルダが起動しシナリオが開きます。
   - エクスプローラで「.scn」ファイルを選択し、シナリオビルダのメイン画面にドラッグアンドドロップします。

注意:

別のシナリオが開いていると、このシナリオを保存してから新規シナリオを開くかどうかを選択するための、ダイアログボックスが表示されます。

---

5 ドキュメントタイプのマッピング

ドキュメントタイプをマップすると、ソースコネクタが生成したドキュメントをターゲットコネクタが取り込めるようになります。シナリオビルダでは、2つのコネクタ間に位置するマッピングボックスで、生成用ソースドキュメントタイプと取り込み用ターゲットドキュメントタイプ間の、マッピングを編集できます。

マッピングを作成するには、ソースドキュメントタイプの要素と、ターゲットドキュメントタイプの要素を関連付ける必要があります。

本章の読解を容易にするために、**ソース要素**や**ターゲット要素**、更にソースまたはターゲットのフィールド、構造体、コレクションの概念を本章で用います。

ソースドキュメントタイプの要素を、ターゲットドキュメントタイプの要素に直接関連付けられない場合は、マッピングスクリプトを作成しなければなりません。

マッピングスクリプトの詳細については、「マッピングスクリプト [p. 109]」の章を参照してください。

マッピングボックス

マッピングボックスでは、ソースコネクタの生成用ドキュメントタイプと、ターゲットコネクタの取り込み用ドキュメントタイプの間のマッピングを編集できます。

マッピングボックスの設定

Connect-Itのマッピングボックスの設定はウィザードで実行できます。

マッピングボックスの設定ウィザードを起動するには、以下の方法があります。

- **マッピングボックスがシナリオ図内に位置しない場合**
  1. ツールボックス内のマッピングボックスをダブルクリックします。
  2. ツールボックスからシナリオ図へマッピングボックスをドラッグします。
- **マッピングボックスがシナリオ図内に位置する場合**
  1. マッピングボックスを選択してから［**ツール／設定**］を選択します。
  2. マッピングボックスを選択し［**F2**］キーを押します。
  3. マッピングボックスを選択し、右クリックしてショートカットメニューから［**コネクタを設定する**］を選択します。
  4. をクリックします。

マッピングボックスの設定ウィザードは2ページから成っています。詳細は次節で説明されています。

注意:

シナリオビルダのツールバーで をクリックしなかった場合は、2番目のページは表示されません。

マッピングボックスに名前を付け、説明を加える

設定ウィザードの1ページ目では、以下の内容を指定できます。

- マッピングボックスの名前
- マッピングボックスの役割の説明

［名前］

マッピングボックスに名前を付けます。フィールドのデフォルト値は「**Mapping**」です。1つの同じシナリオ内でこの名前は固有の名前でなければなりません。

[説明]

コネクタの役割を説明するテキストを作成します。

高度な設定

このページでは、マッピングスクリプト内の演算子[..]の動作を指定できます。Connect-It 3.2より後のバージョンでは、演算子 {..} を [..] の代わりに使用することも可能です。

[..] と {..} 演算子の使用例

以下の表のスクリプトは、FamilyNameフィールドの値が存在するかどうかをテストします。このフィールドに値がある場合、戻り値は常に「Doe」です。

［演算子[..]の適用先になる要素が存在するかどうか確認する］オプションが選択されているか否かによって、またスクリプトで使用されている演算子のタイプに応じて、取り込まれるドキュメントに送信される値は異なります。

重要項目:

空の値に関する概念は、関連するフィールドのタイプに応じて変化します。

- テキストタイプのフィールドの場合、空の値は "" に等しいです。
- 数値型のフィールドの場合、空の値は0に等しいです。
- その他

オプションが選択されていない場合（旧バージョンでの動作）

	FamilyName = ""	FamilyNameは存在しない	FamilyName = "値"
RetVal = "Doe" If [FamilyName] = "" Then RetVal = [FamilyName] End If	空の値	空の値	Doe
RetVal = "Doe" If {FamilyName} = "" Then RetVal = [FamilyName] End If	空の値	空の値	Doe

RetVal = "Doe"
If {FamilyName} = "" Then
  RetVal = {FamilyName}
End If
	空の値	空の値	Doe

オプションが選択されている場合（新規の動作）			

RetVal = "Doe"
If [FamilyName] = "" Then
  RetVal = [FamilyName]
End If
	空の値	スクリプトに関連するノードは、シナリオビルダに拒否されます。メッセージ "" がドキュメントログ内に表示されます。	Doe

RetVal = "Doe"
If {FamilyName} = "" Then
  RetVal = [FamilyName]
End If
	空の値	スクリプトに関連するノードは、シナリオビルダに拒否されます。メッセージ "" がドキュメントログ内に表示されます。	Doe

RetVal = "Doe"
If {FamilyName} = "" Then
  RetVal = {FamilyName}
End If
	空の値	NULLの値	Doe

マッピングの編集

本節では、統合シナリオのマッピングを編集するための手順を説明します。

マッピング編集ウィンドウ

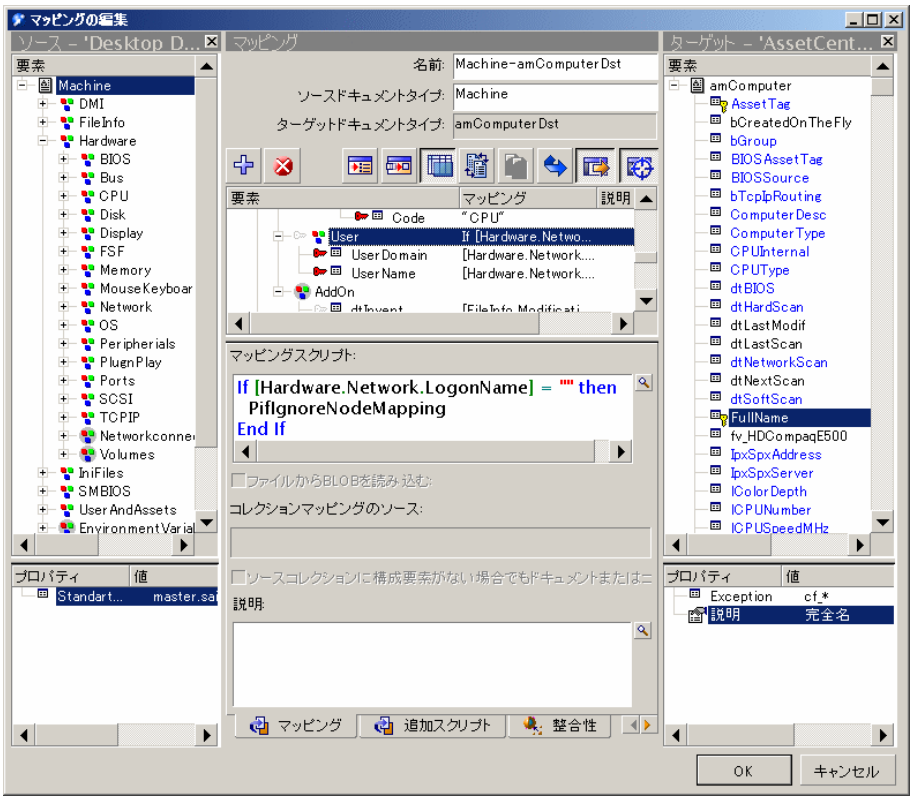
マッピング編集ウィンドウは3つのゾーンに分かれています。

- 選択されたソースコンポーネントが発行する全ての使用可能なドキュメントタイプが表示される部分（画面左）

  ［ソースとターゲットの選択］ボックスで既存の生成用ドキュメントタイプを選択すると、このドキュメントタイプのルートノードの全レベルが表示されます。

- 選択されたターゲットコンポーネントが発行する全ての使用可能なドキュメントタイプが表示される部分（画面右）
- マッピング用ゾーン（画面中央）。ここには以下のゾーンがあります。
  - ソース要素とターゲット要素をマップするための作業枠
  - ターゲットコンポーネントのルール（ディレクティブ）タブと、マッピングの各ノードを説明するためのタブが現れる部分

図 5.1. マッピング編集ウィンドウ



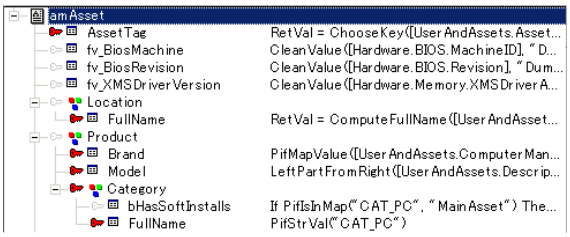
マッピングボックスのツールバーでは、頻繁に行う操作を実行できます。

アイコン	機能
	作業枠内に要素を置きます。
	作業枠内から選択した要素を取り除きます。

アイコン	機能
	ターゲット要素をソース要素に関連付けます。
	マッピングで使用されていないソースとターゲットドキュメントタイプに、フィルタを適用します。
	作業枠内にあるターゲット要素を複製します。
	マップされたターゲット要素を探します。
	ソースコネクタの使用可能なドキュメントタイプを表示／非表示にします。
	ターゲットコネクタの使用可能なドキュメントタイプを表示／非表示にします。
	要素が選択されているゾーン内に、検索用のダイアログボックスを表示します。
	マッピングの作業枠内に、検索と置換用のダイアログボックスを表示します。

マッピングウィンドウの作業枠には、ソース要素とターゲット要素のマッピングの詳細が表示されます。スクリプトが使用されていると、スクリプトの関連付けられているターゲット要素の横に要約された形で表示されます。

図 5.2. 作業枠上のマッピングスクリプトの要約



マッピングボックスには、ソースコネクタとターゲットコネクタが発行する全ての使用可能なドキュメントタイプが表示されています。このため、マッピング編集ウィンドウでマッピングを作成すると（つまりソース要素とターゲット要素を、マッピングボックスで関連付けると）、ソースコネクタの生成用ドキュメントタイプとターゲットコネクタの取り込み用ドキュメントタイプを、マッピングと同時に自動的に作成することができます。

注意:

同じマッピング内で、1つのソースドキュメントタイプの要素を、複数のターゲットドキュメントタイプの要素に関連付けることはできません。またその逆（1つのターゲットドキュメントと複数のソースドキュメント）も不可能です。そのため、マッピングの編集中にドキュメントタイプの1つを変更すると、新規のマッピングを作成することになり、以前に作成したマッピング情報は失われます。1つのソース要素を複数のターゲットドキュメントタイプの要素に関連付けるには、複数のマッピングを作成し、各マッピングに同一のソースドキュメントを使用します。

複数のマッピング用に同じソースドキュメントを使用すると、ソースコネクタ用に実行するクエリの数を最小限に抑えることができます。

また、1つのターゲットドキュメントタイプを選択して、そのドキュメントタイプ内の1つのノードを作業枠内にドラッグした後、ソースドキュメントタイプに関連付けないでおくことも可能です。このノード下にあるフィールドは作業枠内に表示されます。

注意:

ソース要素またはターゲット要素上に表示される**［データを表示する］**ショートカットメニューを使用すると、ソースデータやターゲットデータを表示できます。

ソースドキュメントタイプの枠を表示／非表示にする

1 以下の操作の内1つを実行します。
- をクリックします。
- マッピングの作業枠内で右クリックし、**［ソースを表示する］**を選択／非選択にします。

ターゲットドキュメントタイプの枠を表示／非表示にする

1 以下の操作の内1つを実行します。
- をクリックします。
- マッピングの作業枠内で右クリックし、**［ターゲットを表示する］**を選択／非選択にします。

マッピング編集ウィンドウの各枠を移動させる

1 枠の上端にマウスポインタを置きます。

2 濃淡のバーをクリックすると、枠の周囲を囲む黒線の長方形が表示されます。
3 マウスボタンを押したままにして、希望の位置に黒線の長方形を移動させます。
4 枠を囲む黒線の長方形が希望の位置に移ったら、マウスボタンを放します。

マッピングの作成方法

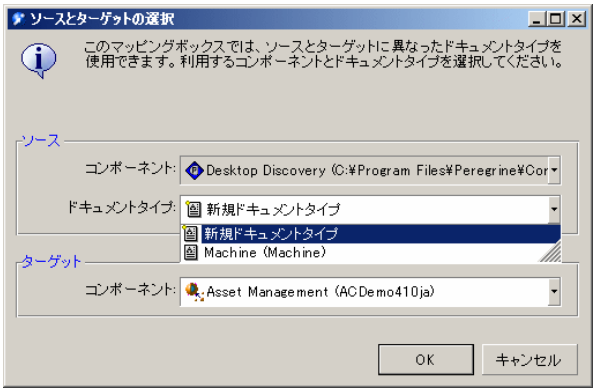
1 シナリオ図内でマッピングボックスを選択します。
2 以下の操作の内1つを実行します。
- **［マッピング］**タブを選択します。
- ダブルクリックします。

  ダブルクリックすると、**［マッピング］**タブと同じ内容のダイアログボックスが表示されます。

3 をクリックします。

［ソースとターゲットの選択］ダイアログボックスが現れます。ここでは以下の要素を選択します。
- ソースコンポーネント
- ソースドキュメントタイプ
- ターゲットコンポーネント

図 5.3. コンポーネントとソースドキュメントタイプの選択用ボックス



注意:

次の場合このダイアログボックスは表示されません。

- マッピングボックスが、1つのソースコンポーネントと1つのターゲットコンポーネントにのみリンクされている場合
- ソースコンポーネント用に、生成用ドキュメントタイプが作成されていない場合

ドキュメントタイプ要素のパス

ドキュメントタイプの要素の場所を［**コレクションマッピングのソース**］フィールドなどに入力するには、この要素のドキュメントタイプ内でのパスを指定する必要があります。ドキュメントタイプの要素の親子関係はピリオドで表現されます。以下の記述は関連し合う要素を表しています。

コレクションXのパスを指定するには以下の様に記述します。

構造体A.構造体B.コレクションX

このシンタックスは、コレクションXは構造体Bの子要素であり、構造体Bは構造体Aの子要素であることを示しています。

例 :

Desktop Discovery - Asset Managementシナリオの「**Asset information**」マッピングでは、ターゲットドキュメントタイプ「**amAsset**」のコレクション「**SubAssets**」は、「**Printer**」コレクションにマップされています。このコレクションは「**Peripherals**」構造体の子に当たり、「Peripherals」はドキュメントタイプ「**Machine**」の「**Hardware**」構造体の子に当たるため、［コレクションマッピングのソース］フィールドには次のパスが表示されます。

Hardware.Peripherals.Printer

要素の名前自体にピリオドが含まれている場合は、シングルクオーテーションマーク（' '）で要素名を囲みます。

'Element.A'

例：

ServiceCenterコネクタの使用可能なドキュメントタイプの要素名の多くは、ピリオドを含んでいます。ServiceCenterコネクタの使用可能なドキュメントタイプ「**ICMDevice**」内にある、「**sw.vendor**」コレクションの［**sw.vendor**］フィールドを指すには、以下の用にパスを記述します。

'sw.vendor'.'sw.vendor'

ピリオドが名前に含まれていない場合、シングルクオーテーションマークはつけてもつけなくても構いません。つまり、

ElementA または 'ElementA'の両方の書き方が可能です。

ドキュメント要素のパス

ドキュメント要素のパスの表記法は、ドキュメントタイプに使用されている表記法と同じです。コレクションの構成要素に関してのみ相違点があります。

ドキュメント内では、コレクションの構成要素は0から番号が付けられています。第1の構成要素には番号0が、第2の構成要素には番号1がついています。

パス内で特定の構成要素を指定するには、以下のシンタックスを用いて構成要素の順位を使用します。

コレクション名(構成要素の順位)

例：**Software**コレクションの第1番目の構成要素の［**VersionName**］フィールドのパスは、以下通りです。

Software(0).VersionName コレクションの第2構成要素用のパスは、Software(1).VersionNameになります。

属性のコレクションのパス

属性のコレクション（1つのフィールドしか含まないコレクション）の特定の構成要素を指すには、パス名に必ず属性の名前を入れます。例えば、**address**属性を含む**Address**コレクションの構成要素を指定するには、Address(1).Address, Address(2).Address, Address(3).Address と記述します。

注意:

コレクションからコレクションへのマッピングを実行する場合は（［**コレクションマッピングのソース**］フィールドに入力します）、コレクションの構成要素の番号を指定する必要はありません。

既存マッピングの変更方法

1 シナリオ図内でマッピングボックスを選択します。
2 以下の操作の内1つを実行します。
- ［マッピング］タブを選択します。
- ダブルクリックします。

  ダブルクリックすると、［マッピング］タブと同じ内容のダイアログボックスが表示されます。

3 以下の操作の内1つを実行します。
- をクリックします。
- 編集するマッピングをダブルクリックします。

マッピングの削除方法

1 シナリオ図内でマッピングボックスを選択します。
2 以下の操作の内1つを実行します。
- ［マッピング］タブを選択します。
- ダブルクリックします。

  ダブルクリックすると、［マッピング］タブと同じ内容のダイアログボックスが表示されます。

3 をクリックします。

マッピングをコピー／貼り付けする

ソースドキュメントタイプからターゲットドキュメントタイプへのマッピング全体、またはマッピングの一部を、クリップボードにコピーします。次に、クリップボードの内容を別のマッピングに追加または結合します。この追加（または結合）先のマッピングでは、コピーされたマッピングのターゲットドキュメントが使用可能である必要があります。（例：Asset Managementコネクタを含むシナリオ内で、ターゲットドキュメントタイプがamAssetになっているマッピングの一部をコピーします。そしてこのコピー内容を、ターゲットドキュメントタイプが同様にamAssetである別のマッピングに貼り付けます。）

この機能は、同一のマッピングまたは類似するマッピングを多数作成する場合に便利です。

1 既存シナリオのマッピングを開きます。
2 マッピングの作業枠内で、ターゲットドキュメントタイプの以下の要素上にポインタを置きます。
- ルートノード

- 構造体
- コレクション
- フィールド

3 右クリックします。
4 **［マッピングをコピーする］**を、表示されるショートカットメニューから選択します。
5 コピーされたマッピングのターゲットドキュメントタイプを含む別のマッピングを開きます（ターゲットドキュメントタイプは、ルートノード、構造体またはコレクションとして含まれています）。
6 作業枠内で、マッピングを追加または結合する場所（ルートノード、構造体、コレクションなど）にカーソルを置きます。
7 右クリックして、ショートカットメニューから**［マッピングを貼り付ける］**を選択します。
8 以下の操作を実行します。
   - **［はい］**をクリックするとマッピングを結合できます。作業枠内で選択された要素は、クリップボード内のマッピングにより変更されます（2つのマッピングが結合し、1つになります）。
   - **［いいえ］**をクリックするとマッピングを追加できます。クリップボード内のマッピングは、既存のマッピングに新規の構造体またはコレクションの形で追加されます。

マッピング説明の記述方法

シナリオビルダではマッピングの各ノードに説明を加えることができます。

例：構造体またはコレクション内に要素が存在するかどうかをテストするために使用されるマッピングスクリプトの説明

1 マッピングを編集します。
2 マッピングのノードを選択します。
3 作業枠の**［マッピング］**タブを選択します。
4 **［説明］**テキストゾーンに説明を加えます。
5 マッピングを保存します。

説明は「.scn」ファイル内に保存されます。

ソースとターゲットのデータをプレビューする

マッピングの編集ウィンドウでは、ソースアプリケーションとターゲットアプリケーションのデータを、プレビュー表示することができます。注意：ターゲットデータのプレビューは一部のコネクタでのみ可能です。

1 マッピングを編集します。
2 ソースまたはターゲットドキュメントタイプのウィンドウ内に、ポインタを置きます。
3 右クリックします。
4 **［データを表示する］**ショートカットメニューを選択します。
データのプレビュー用ウィンドウが表示されます。

データのプレビューを編集する

編集ウィンドウでマッピングのソースドキュメントタイプを編集し、選択した一部のデータのみをプレビューすることも可能です。

1 マッピングを編集します。
2 ソースドキュメントタイプのウィンドウにポインタを置きます。
3 **［データベースのプレビュー用にドキュメントタイプを編集する］**を選択します。
4 マッピングのソースドキュメントタイプの編集ウィンドウが表示されます。
5 作業枠で要素を追加または削除します。
6 をクリックします。
7 ソースデータをプレビュー表示します。
8 **［閉じる］**をクリックして、マッピングの編集ウィンドウに戻ります。
重要：ソースドキュメントタイプを編集するためには、この機能を使用しないでください。
ソースドキュメントタイプに加えられた変更事項は、プレビューウィンドウを閉じると同時に失われます。

複数のマッピングの並べ替え

マッピングは、ソースコネクタの生成用ドキュメントタイプを基準にして並べられています。複数のマッピングが同じドキュメントタイプを含む場合、ドキュメントの生成時にマッピングが実行される順番を指定することも可能です。例えば、最初のマッピングはターゲットアプリケーション内にレコードを作成し、2番目以降のマッピングはこのレコードを更新する、というようにも設定できます。

1 マッピングを選択します。
2 または をクリックして、リスト内でマッピングを上下に移動させます。

ドキュメント生成とマッピング実行の順番

以下の規則が適用されます。

- テストモード

適用されるのは、［生成用ドキュメントタイプ］タブ内に表示されている、ソースコネクタのドキュメント生成の順番です。

- スケジュールモード

  適用されるのは、スケジュールの編集ウィンドウ内で定義されているドキュメントの順番です。このウィンドウでは、1つのスケジューラが複数のドキュメントタイプに関連付けられることがあります。このため、複数のドキュメントタイプがスケジューラに関連付けられる順番が適用されます。この場合、コネクタの［生成用ドキュメントタイプ］タブ内で指定される順番は、考慮されません。

マッピング編集ウィンドウ内で要素を検索する

1. ソースおよびターゲットドキュメントタイプのゾーン、または作業枠内で、1要素を選択します。
2. 以下の操作の内1つを実行します。
   - 右クリックして、ショートカットメニューから［検索（Ctrl＋F）］を選択します。
   -  をクリックします。
3. ［検索する文字列］フィールドに、検索する要素の名前を入力し、以下のオプションを必要に応じて選択します。
   - 大文字と小文字を区別
   - 検索方向（［上へ］または［下へ］）
4. ［検索］をクリックします。

次の要素または前の要素を検索するには、

1. 以下の操作の内1つを実行します。
   - 右クリックして、ショートカットメニューから［次を検索］（「F3」キー）を選択します。
   - 右クリックして、ショートカットメニューから［前を検索］（「Shift＋F3」キー）を選択します。

検索 - 使用上の規則

ドキュメントタイプのツリー構造内での検索は、以下の規則に従います。

- 検索の範囲は、使用可能なドキュメントのウィンドウに表示されている要素に制限されます。サブノード内も検索する場合は、検索の前にサブノードを開いておく必要があります。
- 作業枠内で検索すると、ターゲットドキュメントタイプとマッピングスクリプト内で検索が実行されます。

マッピングスクリプト内で要素を検索し置換する

1 要素を作業枠内で選択します。
2 以下の操作の内1つを実行します。
   - 右クリックして、ショートカットメニューから［**置換え（Ctrl＋H）**］を選択します。
   -  をクリックします。
3 ［**検索する文字列**］フィールドに、検索する要素の名前を入力し、以下のオプションを必要に応じて選択します。
4 ［**置換後の文字列**］フィールドに、この要素の代わりに使用する文字列を入力します。
5 オプションを選択します。
   - 大文字と小文字を区別
   - 検索方向（［上へ］または［下へ］）
6 ［**検索**］、［**置換**］または［**すべて置換**］をクリックします。

マッピングを非アクティブにする

マッピングを非アクティブにすると、コネクタがこのマッピング用にドキュメントを生成するのを妨ぐことができます。例えばあるマッピングは、ソースアプリケーション内で除去された資産に対応するレコードを、ターゲットアプリケーション内で削除できるとします。レコードを1度だけ削除する場合は、レコードの削除が終わった時点でこのマッピングを非アクティブにします。

マッピングを非アクティブにする場合は、［**マッピング**］タブ内で、非アクティブにするマッピングのチェックボックスをオフにします。

マッピングの種類

本節では、シナリオビルダの数種のマッピングについて説明します。

フィールドからフィールドへの直接マッピング

フィールドからフィールドへの直接マッピングを作成するには、ソースフィールドをターゲットフィールドへマップします。データが処理される時、ソースフィールドの値が、マッピングボックスに生成されるドキュメントのターゲットフィールドに入力されます。この操作はBasicエンジンの介入なしに実行されます。

例：

ソースドキュメントタイプの「Name」フィールドは、ターゲットドキュメントタイプの「Name」フィールドに関連付けられます。

フィールドからフィールドへの直接マッピング作成には、2つの方法があります。

1. **ドラッグ&ドロップによる方法**
   - **ソース**または**ターゲット**ウィンドウで、フィールドを1つ選択します。
   - フィールドをターゲットまたはソースウィンドウにドラッグします。
2. **アイコンによる方法**
   - 関連付ける2つのフィールドを、画面のソースゾーンとターゲットゾーンで選択します。
   - をクリックします。
3. **2段階のドラッグ&ドロップ方法**
   - ターゲットフィールドを選択します。
   - 作業枠にフィールドをドラッグします。
   - ソースフィールドを選択します。
   - 作業枠内のターゲットフィールドと同じ行へソースフィールドをドラッグします。
4. **ドラッグ&ドロップと、マッピングスクリプトを使用する方法**
   - ターゲットフィールドを選択します。
   - 作業枠にフィールドをドラッグします。
   - **［マッピングスクリプト］**フィールドに、直接ソースフィールド名を入力します。

     フィールドからフィールドへの直接マッピングをすると、ソース要素名は自動的に［マッピングスクリプト］フィールドに表示されます。手動でソース要素名を入力する場合は、要素名を角括弧（ [] ）で囲む必要があります。

図 5.4. マッピングスクリプト内のソース要素

コレクションからコレクションへのマッピング

コレクションは要素（フィールド、構造体、またはコレクション）から成り立っており、要素が何回繰り返されるかは不定です。

コレクション内要素の1回の繰り返しは、コレクションの構成要素にあたります。

例：Desktop Discoveryコネクタが発行する、使用可能なドキュメントタイプ「Machine」では、あるコンピュータにインストールされたソフトウェアはコレクションで示されています。コレクションの各構成要素はソフトウェアの内容を説明しています（名前、ソフトウェア会社、バージョンなど）。

図 5.5. コレクション

ソースコレクションをターゲットコレクションへマップすると、Connect-Itはデータの処理時に、ソースコレクションにある構成要素の数を計算し、ターゲットコレクション内に同数の構成要素を作成します。

例：

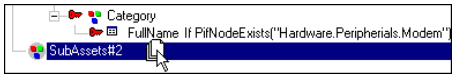
Desktop Discovery - AssetCenterシナリオでは、ドキュメントタイプ「Machine」の「Software」コレクション内の要素は、「amAsset」ドキュメントタイプの「Softinstall」（インストール済みソフト）コレクション内の要素へ関連付けられています。

コレクションからコレクションへのマッピング作成には、2つの方法があります。

- ドラッグ&ドロップによる方法
- キーボードによる方法

ドラッグ&ドロップによる方法

1 ターゲットコレクションを作業枠内にドラッグします。
2 ソースコレクションを選択し、これを作業枠内のターゲットコレクションと同じ行にドラッグします。ドラッグする時には、マウスの左ボタンとキーボードの「Ctrl」キーを同時に押したままにします。
3 マウスポインタ下に3重の文書の形をしたアイコン（コレクション構成要素の重複を表しています）が表示されたら、マウスの左ボタンを放します。



［コレクションマッピングのソース］フィールドには、自動的にソースコレクションのパスが記入されます。

注意:

コレクションの1要素のマッピングを実行するたびに警告が表示され、多くの場合［コレクションマッピングのソース］フィールドに入力する必要があることをユーザに知らせます。［編集／オプション］メニューの［コレクションからコレクションへのマッピングが必要な場合に、警告を表示する］オプション（［確認］項目）で、警告を非表示にできます。

キーボードによる方法

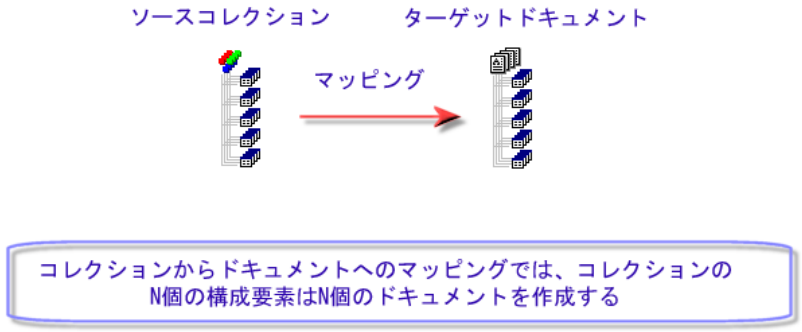
1 作業枠内にターゲットコレクションをドラッグします。
2 ソースコレクションのパス名を［コレクションマッピングのソース］フィールドに直接記入します。このコレクションの値を呼び出すわけではないので、名前は角括弧で囲みません。

コレクションからドキュメントへのマッピング

Connect-Itでは、ソースコレクションをターゲットドキュメントタイプのルートノード（ ）に関連付けることができます。データの処理時に、Connect-Itはソースコレクションにある構成要素と同じ数のターゲットドキュメントを作成します。

コレクションからドキュメントへのマッピングは、コレクションからコレクションへのマッピングと同じ方法で作成されます。ルートノードが、ターゲットコレクションの代わりになります。

図 5.6. コレクションからドキュメントへのマッピング



例：

scac\sc4ac41\scac.scnシナリオでは、「Software」コレクション（コンピュータにインストールされたソフトウェアを記録します）は、「pcsoftware」ドキュメントタイプににマップされています。データの処理時に「Software」コレクションのN個の構成要素は、N個の「pcsoftware」ドキュメントを作成し、これらのドキュメントは更にN個の入力イベントをServiceCenterに送信します。

フィールドからコレクションへのマッピング

フィールドからコレクションへのマッピングでは、コレクションに所属していない1つまたは複数のソースフィールドが、コレクションに所属する1つのターゲットフィールドにマップされます。このターゲットフィールドを含む構造体は、1つの構成要素のみから成るコレクションと見なされます。

例：

［InstalledCards］フィールドは、コンピュータにインストールされた全カード（マザーボード、グラフィックアダプタ、サウンドカード）をまとめています。ソースドキュメントタイプでは、これらの情報が別々のフィールド（［Motherboard］、［GraphicsCard］、［SoundCard］フィールド）になっています。

コレクションを作業枠内にドラッグした後、必要なだけコレクションを複製し、複製したコレクションの［InstalledCards］フィールドをそれぞれ、ソースフィールドにマップします。

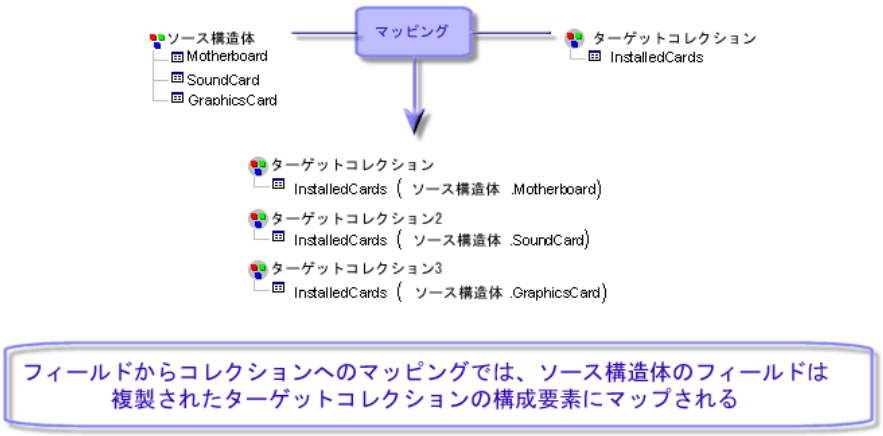
作業枠内のコレクションを複製するには、

1 コレクションを選択します。

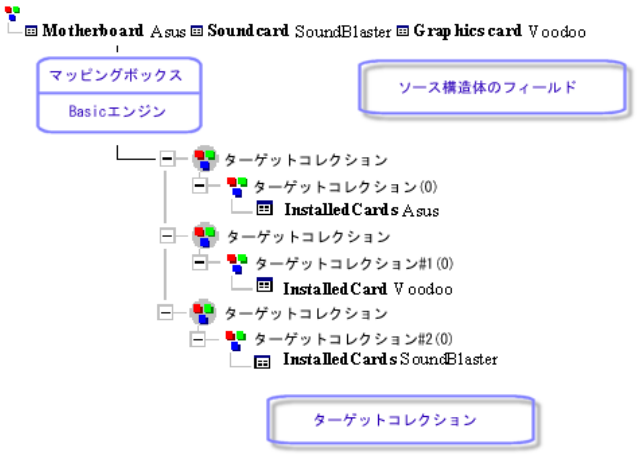
2 をクリックします。

コレクションの1番目の複製には接頭辞#1が付き、2番目には#2が付きます。

図 5.7. フィールドからコレクションへのマッピング



データの処理時に、マッピングボックスは1つの構成要素のみを含むターゲットコレクションを複製された数だけ生成します。各構成要素はそれぞれ1つのソースフィールド値を含みます。



Blobタイプのフィールドのマッピング

Blob（binary large object）タイプのフィールドはバイナリデータ（サウンド、ビデオ、画像ファイル）のフィールドです。

Blobフィールドは別のBlobタイプのフィールドにのみマップされます。

2つの可能性があります。

1. ターゲットBlobフィールドをソースBlobフィールドにマップします。

   この場合バイナリファイルは、シナリオのコネクタを経由してソースアプリケーションからターゲットアプリケーションへ転送されます。

2. Connect-Itとターゲットアプリケーションの共有ファイル内にあるバイナリファイルの名前を指定するフィールドに、ターゲットBlobフィールドをマップします。

   この場合、マッピングスクリプトの入力ゾーン下にある**［ファイルからBLOBを読み込む］**オプションを使用する必要があります。

ファイルからBLOBを読み込む

このオプションでは以下の内容を指定できます。

- BlobファイルはConnect-Itとターゲットアプリケーションの共有フォルダ内にある

  例：[ネットワーク上の共有フォルダ名]/blob/pictures
- ターゲットアプリケーションが共有フォルダからBlobファイルを読み込む

ターゲットアプリケーションが共有フォルダからファイルを読み込むようにするには、

1. マッピング内で、ソースのBlobファイルの名前をターゲットアプリケーションのBlobタイプのファイルへ関連付けます。

   ソースアプリケーションのファイル名は、ソースアプリケーションでBlobファイルに関連付けられている「name」フィールド（icon.name）、または共有フォルダ内のファイルの完全パス（[ネットワーク上の共有フォルダ名]/blob/pictures/monimage.png）になります。
2. **［ファイルからBLOBを読み込む］**オプションを選択します。

   シナリオの起動中に、ターゲットアプリケーションは共有フォルダ内でBlobファイルを取得します。

ソースコレクションに要素がない場合でもドキュメントまたはコレクション要素を作成する

このオプションを選択すると、ソースコネクタに生成される構成要素がコレクションにない場合、以下の操作が実行されます。

- コレクションからドキュメントへのマッピングを実行すると、コレクションは、ターゲットコネクタに取り込まれるドキュメントを作成します。

  本節の「コレクションからドキュメントへのマッピング [p.104]」を参照してください。
- コレクションからコレクションへのマッピングを実行すると、コレクションは、ターゲットコネクタに取り込まれるドキュメント内に、コレクションの1構成要素を作成します。

  本節の「コレクションからコレクションへのマッピング [p.103]」を参照してください。

コレクションの構成要素やドキュメントの作成を強制すると、値がないことをターゲットアプリケーションに通知できるようになります。

6 マッピングスクリプト

フィールドからフィールドへの直接マッピングが不可能な場合、マッピングスクリプト（**［マッピングスクリプト］**フィールド）が必要です。スクリプトは、ソース要素またはデータがターゲット要素に関連付けられるためにどのようにソース要素を操作するか、を指定します。

統合シナリオのマッピングでは、スクリプトで以下の操作を実行できます。

- 固定値をターゲット要素に関連付ける
- 計算された値をターゲットドキュメントタイプのフィールドに関連付ける
- 構造体またはコレクションのフィールドの処理を、条件の検証に従属させる

例：

あるBasicスクリプトは2つのソース要素の連結を可能にします。この連結の戻り値は、データの処理時にターゲット要素に関連付けられます。

プログラム用参考ガイド

スクリプトの作成には、Connect-Itにあるオンラインの『プログラム用参考ガイド』が便利です。

以下の場所にポインタが位置する時に**「F1」**キーを押すと、『プログラム用参考ガイド』が表示されます。

- マッピングの編集ウィンドウ内の［マッピングスクリプト］フィールド
- エディタの入力ゾーン

Basic関数

本節では、マッピングで使用されるBasic命令の一部を説明します。

If、Then、Else、Else If、End If関数

シンタックス

```
If condition Then
 statements
Else If condition Then
 statements
Else
 statements
End If
```

例

```
Dim strVal As String
(...)
If strVal = "" Then
 RetVal = "Empty"
ElseIf strVal = "Default" Then
 RetVal = "Default"
Else
 RetVal = "Unknown"
End If
```

このスクリプトは以下の値を戻します。

- 生成されるドキュメントのテキストフィールドに、情報が含まれていない場合は「Empty」。
- 生成されるドキュメントのテキストフィールドに、「default」情報が含まれている場合は「Default」。
- 生成されるドキュメントのテキストフィールドに、全く別の情報が含まれている場合は「Unknown」。

For Loop関数

この命令ではループを作成できます。

シンタックス

```
For counters = start To end [Step increment]
 statements
Next
```

例

```
For i=0 To 10 Step 2
  PifLogInfoMsg(i)
Next
```

このスクリプトは、ドキュメントログ内に値 (i) を戻します。
ドキュメントログ内での表示内容は以下の通りです。

```
0
2
4
6
8
10
```

While Loop関数

この命令ではループを作成できます。

シンタックス

```
While loop
While condition
  statements
WEnd
```

例

```
Dim i As Integer
i = 0
While i < 10
```

```
  i = i + 2
  PifLogInfoMsg(i)
WEnd
```

このスクリプトは、(i)が10よりも小さい場合に(i)の値をドキュメントログに戻します。

ドキュメントログ内での表示内容は以下の通りです。

```
0
2
4
6
8
10
```

Return関数

このスクリプトでは、この命令の前に定義された条件が満たされていないと、残りのスクリプトは無視されます。

シンタックス

```
(...)
 Return
(...)
```

例

```
If [MacAdress] = "" And [IPAdress] = "" Then
  PifIgnoreNodemapping
  Return
End If

If [MacAdress] <> "" Then
  RetVal = [MacAdress]
Else
  RetVal = [IPAdress]
End If
```

このスクリプトは、生成用ドキュメントタイプ用にMACアドレスとIPアドレスのフィールドをテストします。これらのフィールドが空であると、マッピングノードは無視されスクリプトの最後部は実行されません。

Select関数

この命令は、変数の値に応じて命令のブロックを実行します。

シンタックス

```
Select Case testvar
 Case var1
Statement Block
 Case var2
Statement Block
 Case Else
Statement Block
End Select
```

例

本例の概要

- ソースドキュメントのseStatusフィールドは、チケットのステータスを含んでいます。
- チケットのステータスは以下の通りです。
  - 0 = オープンのチケット
  - 1 = クローズドのチケット

```
Select Case [seStatus]
 Case 0
   RetVal = "Opened"
 Case 1
   RetVal = "Closed"
 Case Else
   RetVal = "Unknown status"
End Select
```

このスクリプトは、チケットのステータスを記述する文字列を、ソースフィールドの数値へ関連付けます。ステータスが不明な場合は、値 "Unknown Status" が戻されます。

PIF関数

PIF関数は、Connect-Itのマッピングスクリプト用に特別に開発されたものです。

全関数は、Connect-Itのインストール先フォルダの「doc」サブフォルダにある、オンライン『プログラム用参考ガイド』内で説明されています。

PifIgnoreDocumentMapping関数

この命令は、ソースドキュメントを無視し、ターゲットコネクタにドキュメントが送信されないようにします。

これは、ドキュメントの複雑な要素（ルートノード、コレクション、構造体、フィールド）全てに適用されます。

シンタックス

```
(...)
 PifIgnoreDocumentMapping("message")
(...)
```

エラーメッセージ（"message"）を追加することも可能です。マッピングボックスの生成用ドキュメントタイプに関するエラーメッセージは、ドキュメントログ内に表示されます。

retval型の命令が指定されている場合、これは、識別キーであるフィールド上にPifIgnore命令が実行される、ということを意味します。

例

```
If [MacAdress] = "" Then
  PifIgnoreDocumentMapping("Missing MAcAdress")
End If

RetVal = [MacAdress]
```

MACアドレスのフィールドは識別キーとして使用されます。このフィールドに値が入力されていない場合、ドキュメントは無視されます。発生するエラー（Missing MacAdress）は、ドキュメントログ内に表示されます。

PifRejectDocumentMapping関数

この命令は、ソースドキュメントを無視し、ターゲットコネクタにドキュメントが送信されないようにします。

これは、ドキュメントの複雑な要素（ルートノード、コレクション、構造体、フィールド）全てに適用されます。

シンタックス

```
(...)
 PifRejectDocumentMapping("message")
(...)
```

エラーメッセージ（"message"）を追加することも可能です。マッピングボックスの生成用ドキュメントタイプに関するエラーメッセージは、ドキュメントログ内に表示されます。

retval型の命令が指定されている場合、これは、識別キーであるフィールド上に**PifIgnore**命令が実行される、ということを意味します。

例

```
If [MacAdress] = "" Then
  PifRejectDocumentMapping("Missing MAcAdress")
End If

RetVal = [MacAdress]
```

MACアドレスのフィールドは識別キーとして使用されます。このフィールドに値が入力されていない場合、ドキュメントは拒否されます。発生するエラー（⚠ Missing MacAdress）は、ドキュメントログ内に表示されます。

PifIgnoreNodeMapping関数

この命令では、ドキュメントタイプの複雑な要素（コレクション、構造体、フィールド）を無視できます。

PifIgnoreNodeMapping命令の動作は、コレクションが命令の対象になっているか否かによって変化します。

この命令がコレクションを対象にしている場合、コレクションの現在の要素のみが無視されます。コレクション全体を無視する場合は、命令**PifIgnoreCollectionMapping**を使用します。

シンタックス

```
(...)
 PifIgnoreNodeMapping("Message")
(...)
```

エラーメッセージ ℹ（"message"）を追加することも可能です。マッピングボックスの生成用ドキュメントタイプに関するエラーメッセージは、ドキュメントログ内に表示されます。

例

```
If [MacAdress] = "" Then
 ' Silently ignoring the element, no string given as parameter of PifIgnoreNodeMapping
 PifIgnoreNodeMapping
End If
RetVal = [MacAdress]
```

このスクリプトにより、MACアドレスを含むフィールドまたは構造体が空である場合に、この空の文字列を使って更新してしまうことを避けることができます。フィールドに値が入力されていると、更新が実行されます。

```
If Left([Software.Name], 7)  = "Windows" Then
 ' If soft name start with "Windows", ignore it
 PifIgnoreNodeMapping
ElseIf Left([Software.Name], 5) = "SunOS" Then
 ' If soft name start with "SunOS", ignore it
 PifIgnoreDocumentMapping
End If
```

このスクリプトにより、"Windows" または "SunOS" を含むデータを無視できます。これらのデータはマップされません。

PifIgnoreCollectionMapping関数

この命令では、コレクションからコレクションへのマッピングを行う際に、生成用ドキュメントタイプの1コレクションを無視できます。

コレクションからコレクションへのマッピングの詳細については、「コレクションからコレクションへのマッピング [p. 103]」の節を参照してください。

シンタックス

```
(...)
 PifIgnoreCollectionMapping
(...)
```

例

```
Dim i As Integer
Dim iCount As Integer

' Number of item in the Logs collection
iCount = PifGetItemCount("Logs")
```

```
' Loop on all logs item to check if there is an error
For i=0 To iCount - 1
 ' Note that it is possible to use the loop index directly in the path to a collection elemen
t ([Logs(i).LogType])
 If [Logs(i).LogType] = 1 Then
   ' LogType value is 1 if it corresponds to an error message.
   ' We have found an item with an error, no need to continue further
   Return
 End If
Next

' If here, means that there is no log item corresponding to an error
PifIgnoreCollectionMapping
```

処理レポート用のこのスクリプトでは、エラーメッセージがない場合は、**logs**コレクションの全構成要素が無視されます。

ドキュメントにエラーがまったくない場合は、このようなスクリプトを使用する必要はありません。**ErrorNumber**フィールドには、ドキュメントに関連したエラーの数が含まれます。

上記のスクリプトの代わりに以下のスクリプトを使用することもできます。

```
If [ErrorNumber] = 0 Then
 PifIgnoreCollectionMapping
End If
```

グローバル変数

マッピングスクリプト内に変数が宣言されている場合、変数の使用範囲はローカルです。

フィールドに関連付けられたスクリプト内で、ある変数を宣言すると、この変数をドキュメントの別の要素へ使用することはできません。

変数の同じ名前が、別の要素で実行されるマッピングスクリプト内で使用されると、新規のローカル変数が作成されます。第1スクリプトの変数の値は、第2スクリプトでは取得できません。

グローバル変数を宣言する

変数の使用範囲がグローバルになるためには、ドキュメントタイプのルートの［**追加スクリプト**］タブで変数を宣言しなければなりません。これによって変数は、同一マッピングの複雑要素に使用されるスクリプトで、計算／使用されるようになります。

注意:

ローカル変数とグローバル変数を識別しやすくするために、グローバル関数に接頭辞「g_」を追加することをお勧めします。

例

```
Dim g_lCounter As Long
```

グローバル変数の宣言は、Basicスクリプトの定義の前に実行されなければなりません。変数はセッションの開始または終了時に使用されます。

例

```
Dim g_lCounter As Long
(...)

Sub OpenSession()
  rem add your code here
End Sub

Sub CloseSession()
  rem add your code here
End Sub
```

ドキュメント数のカウンタを作成する

ドキュメント数のカウンタを作成するには、「グローバル変数 [p. 117]」の節に説明されているように、グローバル変数をまず宣言する必要があります。

この変数のデフォルト値はゼロです。

例

```
g_lCounter = g_lCounter + 1
PifLogInfoMsg(g_lCounter)
```

ソースコネクタがドキュメントを生成するたびに、カウンタは増分され、スクリプトはカウンタの値をドキュメントログに返します。

注意:

シナリオが実行される限り、カウンタは初期化されません。

新規セッションの開始ごとにカウンタ値を初期化するには、追加スクリプトを実行する必要があります。

```
Dim g_lCounter As Long

Sub OpenSession()
  g_lCounter = 0
End Sub
```

ドキュメントタイプのルートにカウンタの増分スクリプト保存することにより、グローバル変数の値は例えば以下のようになります。

```
Session 1
1
2
3
...

Session 2
1
2
3
...
```

ファイルにグローバル変数を保存する

シナリオがサービスモードで実行されている時に停止すると、グローバル変数の現在の値は消失します。

グローバル変数の現在の値を保存するには、ファイルに保存し、必要に応じてこのファイルから読み込まなければなりません。

例

本例の概要

- カウンタは処理されたドキュメント数を数えますが、初期化はされません。
- アプリケーションが停止した場合、カウンタはファイルから読み込まれます。
- ルート要素でグローバル変数が0であると、追加スクリプトは「C:/tmp/counter.txt」ファイルが存在しデータを含むかどうかを確認した上で、ファイルを読み込みます。

  グローバル変数が0であると、これは以下の状況を意味します。
  - シナリオは第1回目に実行された。
  - シナリオは中断された。

```
Dim g_lCounter As Long
```

```
Sub OpenSession()
 ' Counter equals to 0, means that it has not been initialized.
 ' The application has been stopped and we reload the counter
 ' value
 If g_lCounter = 0 Then

  ' If the file does not exist, it must be the first run. Check the
  ' existence of the file to avoid error on opening.
  If FileExists("c:/tmp/counter.txt") Then

   ' Open the file in read mode
   Open "c:/tmp/counter.txt" For Input As #1

   ' Check the file contains data and read the first line of the file.
   If Not Eof(1) Then
    Line Input #1, g_lCounter
   End If
  Close #1
  End If
 End If
End Sub
```

次のスクリプトは、ルート要素上のカウンタが増分されるたびに、ファイル内にカウンタの値を保存します。

```
g_lCounter = g_lCounter + 1
Open "c:/tmp/counter.txt" For Output As #1
Print #1, g_lCounter
Close #1

PifLogInfoMsg(g_lCounter)
```

このスクリプトは、ドキュメントタイプのどの要素にも適用可能です。

マッピングスクリプト作成の手引き

本節では、マッピングスクリプトの作成に役立つ基本概念について説明します。

固定値をターゲット要素に関連付ける

固定値（日付、文字列、数値など）をターゲットフィールドへ関連付けるには、

1 作業枠にターゲットフィールドをドラッグします。

2 ［マッピングスクリプト］フィールドに固定値を入力します。

例：

Desktop Discovery - Asset Managementシナリオでは、スキャンされた各コンピュータは、［amAsset］テーブルに新規の資産を作成します。デフォルトでは、このテーブルの各資産にはカテゴリ名がなければなりません。マッピングでは、固定値 "/Materiel/Ordinateur de bureau/" をamAssetドキュメントタイプの［FullName.Category］フィールドにマップします。

グローバル変数を定義する

マッピングスクリプト用にグローバル変数を定義することができます。

グローバル変数は、ドキュメントタイプのルートに定義された変数であり、ドキュメントタイプの複雑要素（コレクション、構造体、フィールド）に関連付けられたスクリプトに呼び出される変数です。

グローバル変数の宣言は、Basicスクリプトの定義の前に実行されなければなりません。変数はセッションの開始または終了時に使用されます。

グローバル変数は［追加スクリプト］タブで定義されます。

［追加スクリプト］タブ

このタブは、ドキュメントタイプのルート要素でのみ使用可能です。

このタブでは以下の操作を実行できます。

- グローバル変数を定義する
- 各セッションの開始時にBasicコードを実行する
- 各セッションの終了時にBasicコードを実行する

複数のフィールドをドラッグ&ドロップで移動させる方法

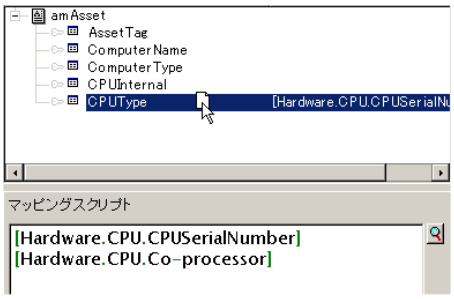
Basicスクリプトでは、Basic関数を用いて複数のソースフィールドを連結させることなどが可能です。

複数のフィールドをドラッグ&ドロップで移動させるには、

1 第1のソースフィールドをターゲットフィールドにマップします。

2 「Shift」キーを押したまま他のフィールドをドラッグします。フィールドはリストとして［マッピングスクリプト］枠内に表示されます。これらのフィールドを使用してスクリプトを作成します。

図 6.1. 複数のフィールドをドラッグ&ドロップで移動させる方法



ソース要素とターゲット要素の位置を見つける方法

作業枠内にあるターゲット要素が、ターゲットドキュメントタイプ内のどこに位置していたかを調べる場合、以下の操作を実行します。

1 要素を作業枠内で選択します。

2 をクリックします。

この操作を行うと、ターゲットドキュメントタイプ枠内にある元のターゲット要素が選択されます。この機能は、ターゲットドキュメントタイプに多数の要素がある場合に便利です。

また、マップされたソース要素（青で表示されています）を作業枠内で見つけることもできます。

見つけるためには、ソースドキュメントタイプ枠内のソース要素をダブルクリックします。するとこのソース要素に関連するターゲット要素が、作業枠内で緑色になります。

ターゲット要素を複製する方法

複数のソース要素に1つのターゲットフィールドを関連付けるためには、ターゲットフィールドを複製する必要があります。例えば、ソースフィールドXとYをあるコレクション内のフィールドAに関連付けるとします。この場合、フィールドAが含まれているコレクションを複製してから、フィールドXを元のコレクションのフィールドAにマップし、フィールドYを複製されたコレクションのフィールドAへマップします。

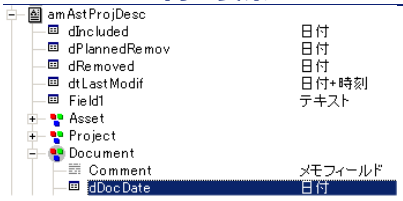
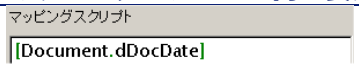
要素を複製するには、

1 作業枠内にある要素を選択します。

2 をクリックします。

複製された要素には番号が付けられます。元の要素には番号0が付いており（これは表示されません）、最初に複製された要素には番号1、2つ目の複製には番号2がついています。この番号の付け方はConnect-Itで強制されており、変更は不可能です。

要素のパスをコピーする方法

ドキュメント内の要素	マッピングスクリプト内の要素のパス
	

マッピングスクリプトでは、使用する要素の完全なパスを指定しなければなりません。例えば上の表では、フィールド［**dDocDate**］の完全パス名は**[Document.dDocDate]**です。

ドキュメントタイプの要素のパスを取得するには、以下の手順に従います。

1. 要素を選択します。
2. 右クリックして、ショートカットメニューから［**パスをコピーする**］を選択します（「**Ctrl+C**」キー）。
3. ポインタを［**マッピングスクリプト**］フィールド内に置きます。
4. 右クリックして、ショートカットメニューから［**貼り付け**］を選択します。（「**Ctrl+V**」キー）

マッピングスクリプトのショートカットメニューを使用する方法

マッピングスクリプト内で右クリックすると、ショートカットメニューが表示されます。

このショートカットメニューでは以下の操作を実行できます（括弧内はショートカットキーです）。

- キー入力を元に戻す
- キー入力を繰り返す
- 切り取り（「**Ctrl+X**」キー）
- コピー（「**Ctrl+C**」キー）

- 貼り付け（「Ctrl+V」キー）
- すべて選択（「Ctrl+A」キー）
- 使用されていないドキュメントタイプにフィルタを適用する
- ソースを表示する
- ターゲットを表示する

マッピング編集ウィンドウのショートカット（コンテキスト）メニュー

マッピングの編集ウィンドウでは、4つのショートカットメニューのコマンドを、マッピングスクリプトを作成する際に使用できます。

ソースコネクタに発行された使用可能なドキュメントタイプの要素上に、ポインタが位置する時

コマンド	機能
このXMLのDTDをコピーする	ソースコネクタの生成用ドキュメントタイプのDTDをコピーします。
パスをコピーする	コンピュータのクリップボードに、選択した要素のパスをコピーします。
選択事項にマップされたノードを表示する	選択された要素に関連する（直接マッピングまたはスクリプト使用のマッピング）マッピングノードを緑色で表示します。
データを表示する	ソースドキュメントタイプのデータを確認するためのウィンドウを表示します。
データベースのプレビューのドキュメントタイプを編集する	コネクタの生成用ドキュメントタイプを作成するためのウィンドウが表示され、生成用ドキュメントタイプのデータを確認できるようになります。
使用されていないドキュメントタイプにフィルタを適用する	現在のマッピングで使用されていないドキュメントタイプにフィルタを適用します。
ソースを表示する	ソースドキュメントタイプ枠を表示／非表示にします。
ターゲットを表示する	ターゲットドキュメントタイプ枠を表示／非表示にします。

ターゲットコネクタに発行された使用可能なドキュメントタイプの要素上に、ポインタが位置する時

コマンド	機能
このXMLのDTDをコピーする	ターゲットコネクタの取り込み用ドキュメントタイプのDTDをコピーします。
パスをコピーする	コンピュータのクリップボードに、選択した要素のパスをコピーします。
この要素を追加する	作業枠内に要素を置きます。
データを表示する	ターゲットドキュメントタイプのデータを確認するためのウィンドウを表示します。
使用されていないドキュメントタイプにフィルタを適用する	現在のマッピングで使用されていないドキュメントタイプにフィルタを適用します。
ソースを表示する	ソースドキュメントタイプ枠を表示／非表示にします。
ターゲットを表示する	ターゲットドキュメントタイプ枠を表示／非表示にします。

作業枠にポインタが位置する時

コマンド	機能
このXMLのDTDをコピーする	ターゲットコネクタの取り込み用ドキュメントタイプのDTDをコピーします。
パスをコピーする	選択した要素のパスをクリップボードにコピーします。
マッピングを編集する	テキストエディタでマッピングスクリプトを編集できます。
マッピングを説明する	選択されたマッピングノードを説明するためのウィンドウを表示します。
［識別キー］	ターゲットコネクタがデータベース型の場合、選択された要素が識別キーとして使用されていることを示します。
この要素を削除する	作業枠内から選択した要素を取り除きます。
マッピングをコピーする	マッピングの一部または全体をクリップボードにコピーします。
マッピングを貼り付ける	現在のマッピング内に、クリップボードのマッピングまたはマッピングの一部を貼り付けます。

コマンド	機能
使用されていないドキュメントタイプにフィルタを適用する	現在のマッピングで使用されていないドキュメントタイプにフィルタを適用します。
ソースを表示する	ソースドキュメントタイプ枠を表示／非表示にします。
ターゲットを表示する	ターゲットドキュメントタイプ枠を表示／非表示にします。

［マッピングスクリプト］の編集用ゾーンにポインタが位置する時

コマンド	機能
キー入力を元に戻す	一番最後のキー入力を取り消します。
キー入力を繰り返す	一番最後のキー入力を繰り返します。
切り取り	選択したテキストを切り取ります。
コピー	クリップボードに選択したテキストをコピーします。
貼り付け	編集用ゾーンにクリップボードの内容を貼り付けます。
全て選択	編集用ゾーンのテキストを全て選択します。
使用されていないドキュメントタイプにフィルタを適用する	現在のマッピングで使用されていないドキュメントタイプにフィルタを適用します。
ソースを表示する	ソースドキュメントタイプ枠を表示／非表示にします。
ターゲットを表示する	ターゲットドキュメントタイプ枠を表示／非表示にします。

文字列

文字列を結合する

演算子&では文字列を結合できます。

文字列でないオペランドが1つでもあると、文字列への変換が自動的に行われます。

例

```
RetVal = "Current date: " & Date()
```

このスクリプトは日付を以下のフォーマットで表示します。

```
Current Date : 2002/12/26
```

文字コードから文字列を作成する

関数Chr()では、Connect-Itが使用するコードページに応じて文字を生成できます。

この関数は、特に以下の型の文字を生成する際に便利です。

- 一重引用符：Chr(39)
- 二重引用符：Chr(34)
- 復帰（キャリッジリターン）：Chr(13)
- 改行（ラインフィード）：Chr(10)

例

```
Dim lVal As Long
lVal = 5
RetVal = "Value: " & Chr(39) & lVal & Chr(39)
```

このスクリプトは以下の値を戻します。

```
Value: '5'
```

UNIX／Windows文字列

UNIXとWindowsで、改行は別の方法で管理されています。

Windowsにおいて改行は文字Chr(13) と Chr(10)で管理されます。UNIXでは文字Chr(10)のみが使用されています。

例

```
RetVal = Replace([WindowsText], Chr(13) & Chr(10), Chr(10))
```

このスクリプトは、ソースドキュメントのWindowsText要素内で取得される文字列内で、Windowsの全ての改行をUNIXの改行に置換します。

関連ファイルの編集

マッピングスクリプトは、場合によって関連ファイルを必要とします。

関連ファイルには以下の要素が含まれます。

- 文字列テーブル

  文字列テーブルの詳細については本章内の節を参照してください。
- マップテーブル

  マップテーブルの詳細については本章内の節を参照してください。
- グローバル関数と変数

  グローバル関数と変数の詳細については本章内の節を参照してください。

これらのファイルは、「.scn」ファイルとは別に、それぞれの内容に対応する拡張子の付いたファイル名で保存されます。

ファイル名の拡張子

.str（文字列用）	文字列テーブル
.mpt（マップテーブル用）	マップテーブル
.bas（Basic用）	グローバル関数と変数

注意:

関連する「.str」、「.mpt」と「.bas」ファイルのないシナリオ（SCNファイル）は機能しません。シナリオを移動させる時は、これらの関連するファイルも同時に移すようにしてください。また、SCNファイルに対してこれらのファイルが適切な場所に位置するようにしてください。ファイルの位置が不適切であると、データの処理時にシナリオがファイルの内容を取得できない可能性があります。

関連ファイルの編集

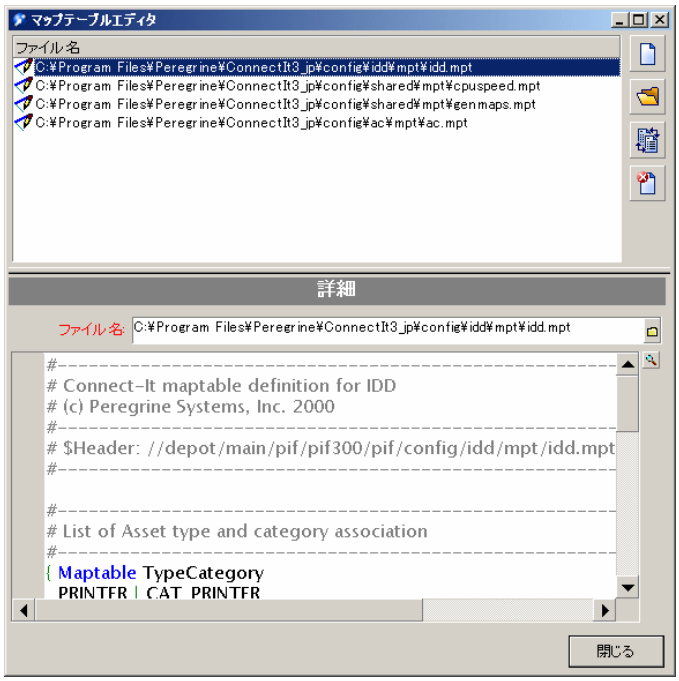
Connect-Itのエディタを使用してこれらの要素を編集します。

エディタを表示するには［シナリオ］メニューを使用します。

［シナリオ］メニューでエディタを選択すると、ウィンドウが開きます。ウィンドウは2つの部分から成っています。

- 現在のシナリオに関連したファイル（「.str」、「.mpt」または「.bas」）を含む枠
- ファイルの編集用の枠（文字列テーブル、マップテーブル、グローバル関数と変数）

図 6.2. マップテーブルのエディタ



上記3つのエディタは、マッピングスクリプトで使用するエディタと同一のテキストエディタを使用します。

新規関連ファイルを作成する

1. ツールバーの をクリックします。
2. **［ファイル名］**フィールドにファイル名を入力します。
   ハードディスク内を検索し、ファイルを作成するフォルダを直接指定するには、 をクリックします。
3. **［作成］**をクリックします。

警告:

ファイルが既に存在する場合、既存ファイルは削除されます。

既存の関連ファイルを開く

既存のファイルを開くには、以下の2通りの方法があります。

関連ファイルの枠（画面上）にあるファイルを開く

現在のシナリオに関連するファイルの枠内にあるファイルを開くには、ファイルをクリックします。

関連ファイルの枠に表示されていないファイルを開く

現在のシナリオに関連するファイルの枠内に、表示されていないファイルを開くには、

1 ツールバーの をクリックします。
2 **［ファイル名］**フィールドにファイルの完全パスを入力します。
ハードディスク内を検索してファイルを見つけることも可能です。
3 **［開く］**をクリックします。

関連ファイルを複製する

1 関連ファイルのウィンドウ内でファイルを選択します。
2 ツールバーの をクリックします。
3 **［ファイル名］**フィールド内に複製されたファイルの名前を入力します。
ハードディスク内を検索し、ファイルを作成するフォルダを直接指定するには、 をクリックします。

関連ファイルを削除する

現在のファイルに関連するファイルを削除するには、

1 関連ファイルのウィンドウ内でファイルを選択します。
2 ツールバーの をクリックします。

スクリプトの認証

スクリプトの一貫性は、Connect-Itによりデフォルトで検査されています。スクリプトの認証では、Basicシンタックスが検査され、また、マッピングスクリプト内で参照されている要素がソースドキュメントに存在するかどうかがテストされます（検証では、ソースドキュメントの構造内で表示されている要素のみが「存在する」ものと見なされます）。

注意:

［コンパイルのエラーを含むスクリプトの認証を許可しない］オプション（**確認**項目）では、上記のスクリプトの認証を有効または無効にできます。

関連ファイルのテキストを編集する

アイコンをクリックすると、テキストエディタにアクセスできます。このアイコンは、各エディタ内と、マッピングの編集ウィンドウ内の［マッピングスクリプト］フィールドの横にあります。

このテキストエディタでは、Basicスクリプトと「.str」、「.mpt」と「.bas」ファイルを容易に編集できます。エディタは通常のテキストエディタの機能（コピー、貼り付け、現在のアクションをキャンセルする、など）を備えています。これらの機能には、［編集］メニューまたはツールバーからアクセスできます。

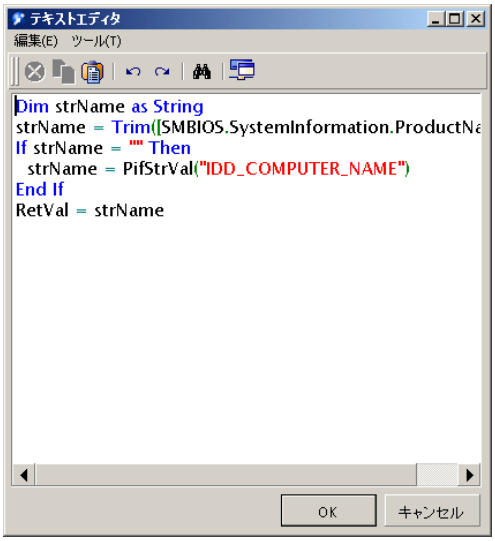


表 6.1. テキストエディタ - ツールバー

アイコン	機能
	選択したテキストを切り取ります。
	テキストをコピーします。
	クリップボードにテキストを貼り付けます。
	現在のアクションをキャンセルします。
	キャンセルしたアクションを繰り返します。
	テキストエディタの設定用ウィンドウを表示します。

テキストエディタの設定

テキストエディタを設定するには、

- をクリックします。

  または

- ［ツール／オプション］メニューを選択します。

テキストエディタを設定すると、ファイル作成用のテンプレートを変更できます。

警告:

各テキストエディタで使用可能なテンプレートを編集する場合は、まず、Connect-Itのインストール先フォルダのサブフォルダ「bin32」にある「codeedit.cfg」ファイルが、読み取り専用でないことを確認してください。

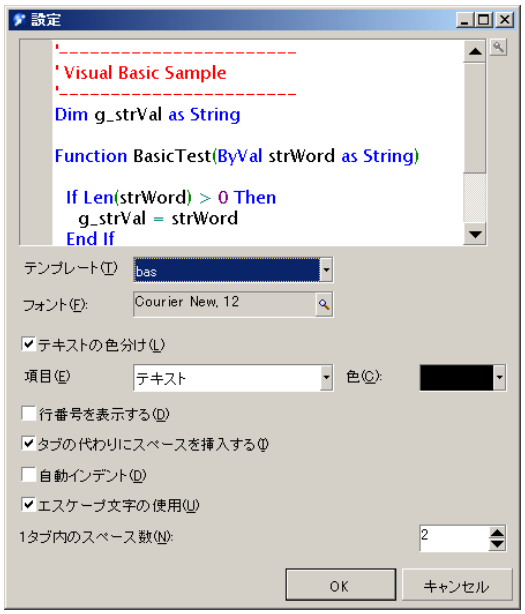
各エディタ用のテンプレートを使用する必要があります。例えばマップテーブルを作成している場合、「mpt」テンプレートがテキストエディタに使用されます。

テンプレート名	編集する要素
bas	マッピングスクリプト グローバル関数
str	文字列テーブル
mpt	マップテーブル
sql	SQLクエリ
scn	Connect-Itシナリオ
Default	上記以外の全ファイル

テンプレートには以下の設定項目があります。

- 文字のフォントを変更する（タイプ、スタイル）
- 要素の色を選択する
- 行番号を表示／非表示にする
- 1から8個のスペースをタブの変わりに挿入する

図 6.3. テキストエディタの設定



テンプレートの変更

テンプレート内で使用されているフォントを変更するには、［フォント］フィールドの横にある拡大鏡アイコンをクリックします。ダイアログボックスが現れ、コンピュータのコンピュータにインストールされているフォントから選択できるようになります。

テキスト内の要素の色も変更できます。色を変更できる要素は以下の通りです。

- テキスト
- 背景
- 選択されたテキスト
- 選択されたテキストの背景
- 行番号
- 数値
- 区切り文字
- コメント
- 文字列
- キーワード
- オペレータ

要素に色をつけるには、［要素］フィールドで1項目を選択し［色］フィールド内で色を選択します。

テキストに色をつけない場合は、［テキストの色分け］チェックボックスをオフにします。

行番号を表示し、タブの代わりにスペースを挿入するには、それぞれのチェックボックスをオンにします。

グローバル関数と変数

［シナリオ］メニューで、グローバル関数と変数のエディタにアクセスします。

マッピングスクリプトは、スクリプト内の数箇所で関数や変数を使用します。これら複数のグローバル関数と変数は、同一の「.bas」ファイル内で保存することができます。保存された後、関数と変数はマッピングスクリプト内で呼び出されます。マッピングボックスは、データの処理中にシナリオに付属した「.bas」ファイルにある関数と変数を使用します。

グローバル関数**CPUSpeed()**は、「gen.bas」ファイルに含まれています。この関数はマップテーブル**CPUspeed()**を参照しており、CPU速度の汎用値を指定します。

```
'------------------------------------------------------------------------------
' Returns the generic CPU speed frequency
' Use the generic maptable
' CPUSpeed'
Function CPUSpeed(ByVal strValue As String) As Integer
  Dim iTmp As Integer
  iTmp = CInt(strValue) / 10
  CPUSpeed = CInt(PifMapValue(CStr(iTmp * 10), "CPUSpeed", 1, strValue))
End Function
```

文字列テーブル

［シナリオ］メニューで文字列テーブルにアクセスします。

文字列テーブルでは、固有の識別子が各文字列に付いています。データの処理時に、Basicスクリプトは識別子を識別子に対応する文字列に置き換えます。

マッピング内で文字列を使用するには、**PifStrVal**関数の後に、ダブルクォーテーションマークと括弧で囲まれた文字列の識別子を入力します。

次の例は、Desktop Discovery - Asset Managementシナリオで使用されている、「category.str」文字列テーブルの一部です。

```
CAT_UNIX, "/Hardware/Unix Workstation"
CAT_SERVER, "/Hardware/Server"
CAT_WORKSTATION, "/Hardware/PC"
CAT_MAC, "/Hardware/Mac"
CAT_TERMINAL, "/Hardware/Terminal"
CAT_PORTABLE, "/Hardware/Portable"
```

例：

"/IT/UNIXワークステーション" の値を取得するには、マッピングスクリプトで次のコード行を使用します：PifStrVal("CAT_UNIX")

このテーブルでは、識別子「CAT_UNIX」は "/IT/UNIXワークステーション" の値に対応しています。このため、文字列内で名前を変更すると、シナリオはこの変更事項を考慮に入れるため、この識別子を使用するBasicスクリプトを再びコンパイルする必要はありません。また、各言語ごとの文字列テーブルを複数使用することもできます。例えば「frcategory.str」をフランス語用に、「grcategory.str」をドイツ語用に使用することなどが可能です。

マップテーブル

［シナリオ］メニューでマップテーブルにアクセスします。

マップテーブルはテーブルの形を取っており、第1列目にはキーが、それ以降の列には値が含まれています。1つの列の中で、各キーは1つの値に対応します。

マッピング内で列の値を取得するには、**PifMapValue()**関数と、キーのパラメータ、キーがマップテーブルで定義されていない場合はデフォルト値、マップテーブルの名前、と列番号を使用します。

データの処理時に、スクリプト内でキーが見つかると、キーはスクリプト内で指定された列の値に置き換えられます。

以下の**Brand**マップテーブルでは、**Apple**キーと**MacIntosh**キーは、スクリプト内で列1番が指定されていると、"Apple" という値を戻します。

```
{ MapTable Brand
Compaq ¦ Compaq
IBM -Lexmark¦ IBM - Lexmark
Hewlett Packard¦ Hewlett Packard
HP-UX¦ Hewlett Packard
Toshiba ¦ Toshiba
Apple ¦ Apple
Macintosh ¦ Apple}
```

例：

"HP-UX" という値を取得するためには、以下のコード行をマッピングスクリプト内で使用します：PifMapValue([マッピングでのフィールド名],"Brand",0,"DefaultValue")

この関数に関する詳しい説明は、オンラインの『プログラム用参考ガイド』の**PifMapValue()**関数を参照してください。

複数の言語用にマップテーブルを作成する

様々な言語の文字列を戻すマップテーブルを作成するには、

1. 文字列ファイルを作成します。ファイル内の各行では、1つの識別子が1つの外国語の文字列（英語、フランス語、など）に、[識別子], ["外国語の文字列"] の形で関連付けられます。例：「category.str」ファイルには、英語用のCAT_UNIX, "UNIX Workstation") という行が含まれており、「fcategory.str」ファイルには、フランス語用のCAT_UNIX, "Station de travail UNIX") という行が含まれています。
2. このファイルをマップテーブルのファイルに含むには、次のシンタックスを使用します：#include_str "[ファイル名]" 例：#include_str "category.str"
3. [dollar]([識別子]) のシンタックスで識別子に各文字列に関連付けると、文字列をマップテーブル内で参照できます。

```
#include_str "category.str"
{MapTable Category
[dollar](IDS_CAT_UNIX)¦ workstation}
{ MapTable Sc2AcCat
 [dollar](IDS_CAT_UNIX) ¦ workstation}
```

ダイナミックマップテーブル

ダイナミックマップテーブルを使用すると、ODBCデータベースでSQLクエリを実行し、マップテーブルと呼ばれるデータテーブルの形でクエリに戻された値を取得することができます。シナリオをテストまたは起動するたびに、このデータテーブルの内容は動的に更新されます。

ダイナミックマップテーブルを作成するには、**PifCreateDynaMaptable**関数を使用する必要があります。この関数に関する詳しい説明は、オンラインの『プログラム用参考ガイド』を参照してください。

ユーザフォーマット

ユーザフォーマットとは、ユーザがマッピングスクリプト内で使用するために定義する日付型または数値のフォーマットを指します。フォーマットは、PifUserFmtVarToStr関数とPifUserFmtStrToVar関数と共にのみ使用されます。使用に関する詳細は、オンラインの『プログラム用参考ガイド』（Connect-Itでのスクリプト作成中に「F1」キーを押すと表示されます）を参照してください。

日付型のユーザフォーマットを作成する

日付型のユーザフォーマットを作成するには、

1. **［シナリオ／ユーザフォーマット］**メニューを選択します。
2. 表示されるウィザードのページで、**［次へ］**をクリックします。
   このページに表示される日付型のフォーマットを、ユーザが定義することはできません。
3. 表示されるウィザードのページで、 をクリックします。
4. 既存フォーマットのリスト内で、リスト最後のフォーマット名の下（**［名前］**の列）をクリックします。
5. 表示される編集用ゾーンにフォーマットの名前を入力します。
   名前にスペースを使用することはできません。
6. 入力した名前の隣の**［フォーマット］**列をクリックします。
7. **［日付型フォーマットの記号］**枠内の属性を用いて、公式を入力します。
8. をクリックします。
   作成したフォーマットの例が、**［日付型フォーマットの結果］**フィールドに表示されます。

数値のユーザフォーマットを作成する

数値のユーザフォーマットを作成するには、

1. **［シナリオ／ユーザフォーマット］**メニューを選択します。
2. 表示されるウィザードのページで、**［次へ］**をクリックします。
   このページに表示される数値のフォーマットを、ユーザが定義することはできません。
3. をクリックします。
4. 既存フォーマットのリスト内で、リスト最後のフォーマット名の下（**［名前］**の列）をクリックします。
5. 表示される編集用ゾーンにフォーマットの名前を入力します。

名前にスペースを使用することはできません。

6 入力した名前の隣の［**フォーマット**］列をクリックします。

7 ［**数値フォーマットの記号**］枠内の属性を用いて、公式を入力します。

8 ✓をクリックします。

作成したフォーマットの例が、［**数値フォーマットの結果**］フィールドに表示されます。

フォーマット作成に使用するシンタックス

フォーマット公式の作成時には、以下の規則を守る必要があります。

- 日付型または数値のフォーマットに使用する記号は、Windowsオペレーティングシステムに使用されている記号と同一にします。
- フォーマット内に現れる文字列は、シングルクォーテーションマークで囲みます。
- 2つの値の間のスペースも、文字列と同様にシングルクォーテーションマークで囲みます。

表 6.2. 日付型フォーマットの例

フォーマットの公式	例
yyyy'-'mm'-'dd	2002-02-07
hh':'nn':'ss	11:55:29
h':'nn':'ss	11:55:41
hh':'nn	14:18
hh'' 'h' ''nn	15:54:53
h t' 'nn' 'tt	2 29 pm
dd'/'MM'/'yy	07/02/02
dd'.'MM'.'yy	07.02.02
dd'-'MM'-'yy	07-02-02
dd'/'MM'/'yyyy	07/02/2002
yyyy'年'MM'月'dd'日'dddd	2002年02月07日木曜日

表 6.3. 数値のユーザフォーマットの例

フォーマットの公式	例
n' 'n','dd-'USD'	1 0 2 0 3,41-USD
-'USD'n'.'nnn','ddd	-USD10.203,408
-n'.'nnn','dd'USD'	-10.203,41USD

フォーマットの公式	例
-n'.'nnn','ddd'USD'	-10.203,408USD

コレクション - 例

本節では、ドキュメントタイプの複雑要素の一つであるコレクションを対象とするスクリプトの例を、幾つか紹介します。

値のリストに基づいてコレクション内に構成要素を作成する

本節では、ソースドキュメントの値のリストに基づいて、あるコレクション内に構成要素を作成するスクリプトの例を説明します。

本例の概要

- **Software**ソースフィールドは、値のリストを含んでいます。
- 値は区切り文字で区切られています。

スクリプトの役割

- ソフトウェア名を1つずつ抽出します。
- ターゲットコレクション**SoftInstalled**内に1構成要素を作成します。
- 抽出されたソフトウェア名を使って、**Name**要素に値を入力します。

```
Dim iCount  As Integer
Dim iIndex  As Integer
Dim strSoft As String
Dim lDummy  As Long
Dim strPath As String

' ソースフィールド "Software" 内で値の数を数えます。
' ソフトウェア名はカンマの（','）で区切られます。例："Excel, Connect-It,
' AssetCenter"
iCount = CountValues([Software], ",")

' リストの全要素から1つずつ抽出するために、全要素上で巡回して繰り返します。
For iIndex = 0 To iCount - 1

 strSoft = GetListItem([Software], ",", iIndex+1)

 ' ソフトウェア名前後のスペースの削除
```

```
 strSoft = Trim(strSoft)

 ' ルート要素からターゲットコレクションのパスを作成。
 ' 例えば、第3番目のソフトウェアにはパス "SoftInstalled(3).Name" が作成されます。
 strPath = "SoftInstalled" & iIndex & ").Name"

 ' ソフトウェアの文字列タイプの現在の値を、以下を使用してこのパスに割り当てます。
 ' la fonction PifSetStringVal.
 ' La fonction PifSetStringVal パスが有効でない場合はエラーコードが返されます。
 ' 変数に関数の戻り値を割り当てる必要があります。割り当てられていないと
 ' 関数は適用されません。
 lDummy = PifSetStringVal(strPath, strSoft)

Next iIndex
```

このマッピングスクリプトは、ターゲットドキュメントタイプのどの要素にも適用可能です。

マッピングを読みやすくするために、構成要素の追加先のコレクションに、このスクリプトを実行することをお勧めします。

警告:

Basic関数**PifSetStringVal**の呼び出し内にパスで指定されている要素は、ターゲットドキュメントタイプ内に存在しなければなりません。現在の例では、ユーザは**SoftInstalled**コレクションの**Name**要素を、取り込み用ドキュメントタイプ内に追加しなければなりません。

コレクションの複数の構成要素を1フィールド内に結合する

本例の概要

- ソースドキュメントは、値のコレクションを含んでいます。
- このコレクションの要素は、ターゲットドキュメントタイプのフィールドへマップされています。

ソースは、あるコンピュータにインストールされたソフトウェアのコレクションを含んでいます。ソフトウェアの名前は、カンマ（','）で区切られたソフトウェアのリストを含むフィールド内に書き込まれなければなりません。

```
Dim iCollectionCount As Integer
iCollectionCount = PifGetItemCount("SoftInstalled")

Dim strList As String
Dim iItem As Integer

各コレクションの要素ごとに、ソフトウェアの名前を取得し、（"SoftInstalled"
コレクションの "Name" 要素）現在のリストに結合します。
For iItem = 0 to iCollectionCount - 1

 ' リストが空でない場合は名前の区切り文字を追加します。
 If strList = "" Then
   strList = strList & ", "

 ' 現在のリストにソフトウェア名を追加します。
 ' コレクション内の構成要素の番号を指定するには、直接変数を使用できます
。
 ' 例えばiITemの変数の値が3である場合、パス
 ' [SoftInstalled(3).Name] は、iItemの値から自動的に作成されます。
? strList = strList & ", " [SoftInstalled(iItem).Name]
Next iItem

' 変数strListをターゲット要素に割り当てます。
RetVal = strList
```

コレクション内で複数のフィールドをマップする

本例の概要

- ソースドキュメントには複数の異なったフィールドが含まれています。

  フィールド［Address1］と［Address2］は、クライアントの2つのアドレスを含んでいます。
- これらのフィールド値は、ターゲットコレクションの構成要素に関連付けられなければなりません。

  本例ではターゲットコレクションは**Address**です。

例

以下の操作を実行しなければなりません。

- ターゲットコレクション内に2つの構成要素を作成し、"Adress1" と "Adress2" フィールドに関連付ける。
- コレクションの複製機能を使用する。
  1. ターゲットドキュメントタイプ内に**Address**コレクションを追加します。
  2. このコレクションを複製します。

Adress#1コレクションが、ターゲットドキュメントタイプ内に表示されます。

3 マッピングスクリプト［Adress1］と［Adress2］は、それぞれAdress.AdressとAdress#1.Adressフィールドに適用されます。

コレクションからコレクションへのマッピングで、コレクションの一部の構成要素を無視する

コレクションの一部の構成要素を無視するには、PifIgnoreCollectionMappingとPifIgnoreNodeMapping命令を使用しなければなりません。

これらの命令の詳細については「PifIgnoreCollectionMapping関数 [p. 116]」の節を参照してください。

7 ピボットドキュメントタイプ

ピボットドキュメントタイプは、構成要素（資産、ソフトウェア、従業員など）の一般的な表現に当たります。ソースコネクタは、自分のドキュメントタイプとピボットドキュメントタイプ（生成用と取り込み用）を対応させます。

ピボットドキュメントタイプを使用するコネクタ同士は、マッピングを定義せずに関連付けられます。

ピボットドキュメントタイプは以下のコネクタで使用できます。

- Asset Managementコネクタ（4.xxと4.1バージョン）
- ServiceCenterコネクタ（3.xxと4.xxバージョン）
- InfraTools Desktop Discoveryコネクタ
- InfraTools Managementコネクタ
- Unicenter AMOコネクタ
- PDA inventoryコネクタ
- Tivoli Inventoryコネクタ
- Winpark Actimaコネクタ
- ゲートウェイ3.xコネクタ

ピボットドキュメントタイプの機能

ピボットドキュメントタイプは、一般的なドキュメントタイプです。

ソースコネクタは、使用可能なドキュメントタイプからピボットドキュメントタイプへのマッピングを定義します。ターゲットコネクタはピボットドキュメントタイプから、取り込み用ドキュメントタイプへのマッピングを作成します。これにより、特定のマッピングを作成せずに、ソースコネクタのデータをターゲットコネクタへ送信することが可能になります。ピボットドキュメントタイプからまたはピボットドキュメントタイプへのマッピングでは、ソースコネクタとターゲットコネクタ間でデータを交換するために必要なデータ変換を定義します。

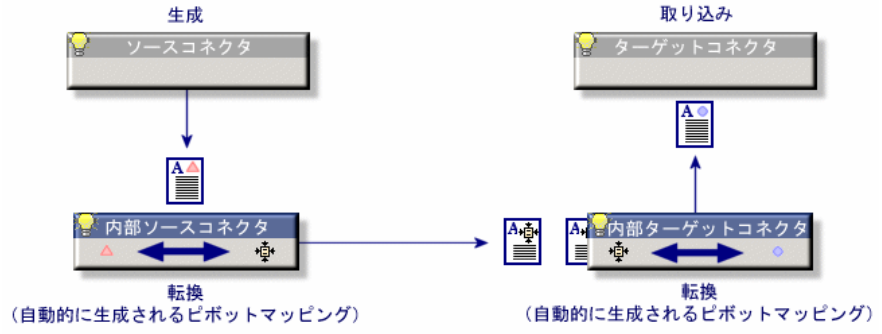
自動的に作成されるマッピング

ソースコネクタが、ピボットドキュメントタイプに基づいて初めてドキュメントを生成すると、自動的に以下のマッピングが作成されます。

- ソースコネクタの生成用ドキュメントタイプと、ピボットドキュメントタイプ間のマッピング
- ピボットドキュメントタイプと、ターゲットコネクタの取り込み用ドキュメントタイプ間のマッピング

以上2つのマッピングはピボットマッピングに基づいています。ピボットマッピングは、それぞれの使用可能なドキュメントタイプをピボットドキュメントタイプにマップします。

図 7.1. ピボットドキュメントタイプの使用



ピボットドキュメントタイプを使用して統合シナリオを作成する

本節では、ピボットドキュメントタイプを使用して統合シナリオを作成する方法を説明します。このシナリオは、1つのソースコネクタと1つのターゲットコネクタを使用します。

シナリオを作成するには、

1 シナリオビルダを起動します。
2 ソースコネクタとターゲットコネクタを、シナリオ図内にドラッグします。
3 **［コネクタの設定］**ウィザードで2つのコネクタを設定します。

［コネクタの設定］ウィザードの**［ピボットドキュメントタイプの使用］**ページで、**［ピボットドキュメントタイプを使用する］**オプションを選択します。

4 **「Shift」**キーを押したまま、ソースコネクタをターゲットコネクタにリンクします。

「Shift」キーを押したままにしないと、2つのコネクタ間に自動的にマッピングボックスが作成されます。

5 ソースコネクタを選択します。
6 ソースコネクタの使用可能なドキュメントタイプのリストにある、ピボットドキュメントタイプ（PrgnAssetなど）に基づいて、1つまたは複数の生成用ドキュメントタイプを作成します。
7 ソースコネクタを起動してシナリオをテストします。

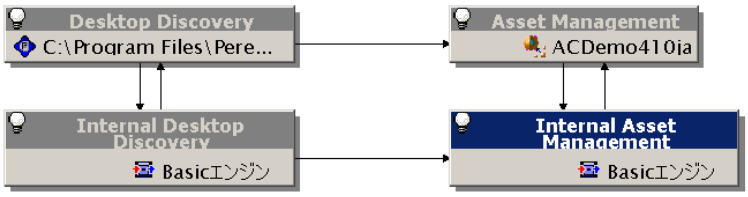
ソースコネクタを初めて起動すると、使用可能なドキュメントタイプとピボットドキュメントタイプ間のマッピングが、自動的に作成されます。

8 スケジュールとサービスをシナリオに関連付け、シナリオをプロダクションモードにします。

内部コネクタを表示する

コネクタがピボットドキュメントタイプを使用すると、シナリオの各コネクタ用に内部コネクタが作成されます。内部コネクタを表示するには、**［シナリオ／内部コネクタを表示する］**メニューを選択します。

図 7.2. 内部コネクタの表示



内部コネクタを表示すると、ピボットドキュメントタイプが使用される時に自動的に作成されるマッピングを、編集できるようになります。

[マッピング] タブと [ピボットマッピング] タブ

ピボットコネクタをシナリオ図内で選択すると、2つのタブが表示されます。

- [マッピング] タブ
- [ピボットマッピング] タブ

[マッピング] タブ

このタブは、コネクタの生成用または取り込み用ドキュメントタイプと、ピボットドキュメントタイプ間に自動的に作成されたマッピング用です。

これらのマッピングを参照することは可能ですが、シナリオが起動される度に再生成されるため、編集は不可能です。マッピングを変更するには、その土台にあたるピボットマッピングを変更する必要があります。

[ピボットマッピング] タブ

このタブは、コネクタの使用可能なドキュメントタイプと、ピボットドキュメントタイプ間のマッピング用です。これらのマッピングは、自動的に作成される **[マッピング]** タブ内のマッピングの基礎となる、モデルの役割を果たしています。

ピボットドキュメントタイプのリスト

ピボットドキュメントタイプには以下のものがあります。

- **prgnAsset**
- **prgnCompany**

- prgnContract
- prgnCostCenter
- prgnDepartment
- prgnFeatures
- prgnLicense
- prgnLocation
- prgnModel
- prgnNetworkDevice
- prgnPC
- prgnPerson
- prgnPrinter
- prgnSoftInst
- prgnTelephony

コネクタの使用可能なドキュメントタイプのリスト内では、この種のドキュメントタイプは専用のアイコン（）で示されています。

「使用可能なドキュメントタイプ」と「ピボットドキュメントタイプ」間のマッピングを変更する

コネクタの使用可能なドキュメントタイプと、ピボットドキュメントタイプ間のマッピングを変更するには、

1 ピボットドキュメントタイプを使用するシナリオを作成します。
2 **［シナリオ／内部コネクタを表示する］**メニューで、ピボットコネクタを表示します。
3 ピボットコネクタを選択します。
4 **［ピボットマッピング］**タブでマッピングを選択します。
5 マッピングを変更します。

コネクタの使用可能なドキュメントタイプと、ピボットドキュメントタイプ間のマッピングの詳細情報については、ピボットドキュメントタイプ用各コネクタの**「ピボットマッピング」**節内を参照してください。

注意:

マッピングを変更するには、コネクタの「.piv」ファイルが読み取りと書き込み可能でなければなりません。

ターゲットコネクタを変更する

ピボットドキュメントタイプを使用すると、シナリオ内でターゲットコネクタを変更しても、他に変更を加えずに、ソースコネクタのドキュメント生成をそのまま再起動することができます。

ターゲットコネクタを変更するには、

1. ピボットドキュメントタイプを使用したシナリオを開きます。
2. シナリオ図内に新規のターゲットコネクタを置きます。
3. **［コネクタの設定］**ウィザードで、コネクタを設定します。
4. **「Shift」**キーを押したまま、ソースコネクタをターゲットコネクタにリンクします。
5. スケジュールまたはソースコネクタを手動で再起動します。

8 統合シナリオのテストとデバッグ

シナリオをテストすると以下の内容を確認できるようになります。

- 生成用ドキュメントタイプを作成した場合、ソースコネクタが適切にドキュメントを生成するか
- マッピングボックスがこれらのドキュメントを適切に変換するか
- マッピングボックスがドキュメントを変換した後、ターゲットコネクタがドキュメントを適切に取り込むか

警告:

テストには、テスト用データ（データベースまたはデモ用ファイル）の使用をお勧めします。テストをせずにシナリオをプロダクションモードで使用すると、実際のデータを破損する可能性があります。

生成用ドキュメントタイプのテスト

1つのシナリオ内でコネクタが複数の生成用ドキュメントタイプを使用する場合、各生成用ドキュメントタイプをテストすることをお勧めします。

ソースコネクタの生成用ドキュメントタイプをテストするには、

1 ソースコネクタを開きます。

2 ［ドキュメントタイプ］タブを選択します。
3 テストするドキュメントタイプ以外の、生成用ドキュメントタイプのチェックボックスを全てオフにします。

複数のドキュメントタイプをテストする場合は、ドキュメントタイプを複数選択したまま次の手順に進みます。

4 ▶をクリックして、または［**ツール／生成する**］メニューを選択してドキュメントタイプの生成を起動します。
5 ドキュメントログの内容を読み、ソースコネクタがドキュメントを生成する際にどのような問題が発生したかを確認します。

注意:

シナリオビルダのオプションの、（［編集／オプション］）［**ドキュメントの生成テスト中に生成するドキュメントの数**］オプションで、テスト時に生成するドキュメントの数を指定できます。

このオプションは、生成用ドキュメントタイプをテストする際に自動的に使用されます。しかし、シナリオがWindowsのサービスとして実行されると、このオプションは自動的に無効になります。

ドキュメントログの使用

ドキュメントログを使用すると、ソースコネクタがドキュメントを生成している際にどのような問題が発生したかを確認できます。

問題を解決するには、処理中に問題の起こったドキュメントのみを表示するようドキュメントログを設定することをお勧めします。

ドキュメントログに関する詳細は、「シナリオビルダ [p. 27]」章の「ログ [p. 49]」の節、「ドキュメントログ [p. 51]」を参照してください。

キャッシュファイルを使用する

コネクタ用のキャッシュファイルには、コネクタの使用可能なドキュメントタイプの情報が含まれています。

例 :

データベース型のコネクタを開くと、コネクタは設定で指定したデータベースの全テーブルの詳細を取得します。キャッシュファイルを使用すると、最後にコネクタを開いた際に作成されたキャッシュファイル内のデータ記述を使用するため、データ記述を新たに取得する必要がなくなります。

シナリオのテストやデバック段階でシナリオのコネクタを頻繁に開閉する場合、この機能は非常に便利です。

キャッシュファイルは以下のコネクタで使用できます。

- Action Request Systemコネクタ
- Asset Managementコネクタ
- ServiceCenterコネクタ
- InfraTools Network Discoveryコネクタ
- InfraTools Managementコネクタ
- FacilityCenterコネクタ
- TeleCenterコネクタ
- InfraTools Managementコネクタ
- LDAPコネクタ
- データベースコネクタ
- Lotus Notesコネクタ
- 全てのインベントリコネクタ

コネクタが外部アプリケーションに接続せずに、キャッシュファイルを使用するように設定するには、

1 **［コネクタの設定］**ウィザードの**［キャッシュを設定する］**ページで、**［キャッシュファイルを使用する］**を選択します。
2 コネクタの使用可能なドキュメントタイプの詳細を、キャッシュファイルに含めるために、コネクタを開きます。

  注意：コネクタを最初に開く際に、キャッシュの使用を示すアイコンは、コネクタのボックス上には表示されません。
3 コネクタを一旦閉じてから、再び開きます。

  この時点で、コネクタは外部アプリケーションには接続せず、キャッシュファイルから使用可能なドキュメントタイプを発行します。

注意:

キャッシュファイルを使用するコネクタを開くと、黄色の電球のアイコンの上に、データベースを表すアイコンが表示されます。

図 8.1. コネクタ - キャッシュ使用を示すアイコン

キャッシュファイルの内容を消去する

キャッシュファイルの内容を消去するには、

1 コネクタを開きます。

2 **［シナリオ／キャッシュを削除する］**メニューを選択します。

コネクタ1つのみのキャッシュファイルの内容を削除するには、［ツール／キャッシュ／キャッシュを削除する］を選択します。

コネクタを次回開くと、コネクタは外部アプリケーションに接続し、キャッシュファイルの内容と外部アプリケーション内の情報の同期をとります。

警告:

キャッシュファイルの同期をとるには、シナリオビルダのオプションで［オフラインで作業する］を［いいえ］にする必要があります。

キャッシュファイルの内容と外部アプリケーション内の内容の同期をとる

キャッシュファイルの内容と、外部アプリケーション内の内容の同期をとるには、

1 シナリオのコネクタを開きます。

2 **［シナリオ／キャッシュの同期をとる］**メニューを選択します。

コネクタ1つのみのキャッシュファイルの同期をとるには、［ツール／キャッシュ／キャッシュの同期をとる］を選択します。

3 Connect-Itがコネクタの使用可能なドキュメントタイプを取得するのを待ちます。

同期化が終了すると、コネクタの発行する使用可能なドキュメントタイプは、外部アプリケーションの内容に一致するようになります。

警告:

キャッシュファイルの同期をとるには、シナリオビルダのオプションで［オフラインで作業する］を［いいえ］にする必要があります。

オフラインで作業する

オフラインモードを使用すると、シナリオのコネクタを外部アプリケーションに接続せずに作業することができます。

オフラインモードを使うには、コネクタを1回目に開いた後に をクリックします。

オフラインモードは、シナリオのコネクタ用にキャッシュファイルを自動的に作成します。

コネクタを閉じた後に新たにコネクタを開くと、オフラインモードが有効になっている場合、コネクタはキャッシュファイルの内容に基づいて、使用可能なドキュメントタイプを発行します。

注意:

オフラインモードの使用中にコネクタを開くと、消えた電球のアイコンの上に、データベースを表すアイコンが表示されます。

図 8.2. コネクタ - オフラインセッションを示すアイコン

重要項目:

オフラインモードでは、ソースからデータを取得することはできません。また、生成用ドキュメントタイプの編集ウィンドウでのソースデータのプレビュー機能も、使用できません。このモードはマッピングを定義するためのみに使用されます。

1コネクタの始動時に現在のシナリオの全コネクタを開く

このオプションにより、1つのコネクタの起動時に全コネクタが自動的に開くように設定できます。このオプションは、主に多数のコネクタを使用するシナリオで使用されます。

テストモード

- ターゲットコネクタがトランザクションをサポートする場合、データが挿入されても、トランザクションのロールバック（データ挿入前の状態への回帰）が実行されます。これにより、ターゲットアプリケーションのデータを

変更することなく、ドキュメントの取り込みをテストできるようになります。

- ターゲットコネクタがトランザクションをサポートしない場合、データはターゲットアプリケーションへは送信されません。このためドキュメントの生成とマッピングのテストは実行できますが、ドキュメントの取り込みをテストすることはできません。

一時停止を実行する

シナリオ内で多数のドキュメントを処理する場合、全ソースデータの処理の終了を待たずにConnect-Itログを読み、エラーがあるかどうかを確認するには、一時停止を実行する必要があります。

マッピング定義に関するアドバイス

Basicスクリプトのシンタックスエラーの場所（マッピング、関連する要素）を見つけるのが困難な場合もあります。このため、全スクリプトを作成してまとめてテストするよりも、1スクリプトの作成ごとにテストを実行することをお勧めします。

9 シナリオ文書

シナリオビルダでは、現在のシナリオに関する詳細を含むシナリオ文書を表示または作成することができます。

シナリオ文書の情報はシナリオのSCNファイルから動的に抽出されるため、常に更新された内容になっています。

シナリオ文書の内容

文書には以下のセクションが含まれています。

- 一般情報

  このセクションには以下の情報が含まれています。

  - 名前

    シナリオ名の前にはコンピュータ上の完全パスがあります。
  - 最終変更日
  - シナリオの最終変更時に使用したConnect-Itのバージョンとビルド番号
  - シナリオの最終履歴にユーザが入力した全情報（**［ファイル／シナリオの履歴］**メニュー）。
- シナリオ図に関する情報

  このセクションには以下の情報が含まれています。

- ビューに関する情報
  各ビューに関連するマッピングが列挙されます。
- ビューに関連付けられた図
  「.bmp」形式の画像で、シナリオのビューのプレビューが表示されます。

- コネクタの設定
  このセクションには、シナリオのコネクタの設定パラメータが含まれています。コネクタの設定に関する詳細については、マニュアル『**コネクタ**』の**「コネクタの設定」**の章を参照してください。
- ドキュメントタイプごとのWhere句とOrder by句のリスト
  このセクションには、ドキュメントタイプごとにWHERE句とORDER BY句が列挙されています。これらの句は、シナリオのソースコネクタの生成用ルールの一部を成します。
- ピボットドキュメントタイプ
  このセクションにはシナリオの各ピボットドキュメントタイプの要素が含まれます。
  ピボットドキュメントタイプの詳細については、「ピボットドキュメントタイプ [p. 143]」の章を参照してください。
  各ピボットドキュメントタイプごとに以下の情報が表示されます。
  - ピボットドキュメントタイプ名
  - 各要素の名前
  - 要素の属性
    - フィールド（ATTRIBUTE）
    - 構造体（STRUCT）
    - コレクション（ARRAY）
  - フィールドのデータ型：String、Short、Double、Timestamp、など

  注意:
  シナリオにピボットドキュメントタイプが含まれていない場合、このセクションは表示されません。

- ユーザ定義のドキュメントタイプ
  このセクションはユーザが定義する各ドキュメントタイプの要素をまとめています。
  各ドキュメントタイプごとに以下の情報が表示されます。
  - ドキュメントタイプ名
  - 各要素の名前
  - 要素の属性
    - フィールド（ATTRIBUTE）

    - 構造体（STRUCT）
    - コレクション（ARRAY）
  - フィールドのデータ型：String、Short、Double、Timestamp、など

  ドキュメントタイプの作成に関する詳細は、「統合シナリオのインプリメンテーション [p. 71]」章の「生成用または取り込み用ドキュメントタイプを編集する [p. 76]」の節を参照してください。
- ピボットマッピング

  このセクションには、シナリオのピボットマッピングの詳細が含まれています。

  注意:

  シナリオにピボットマッピングが定義されていない場合、このセクションは表示されません。

- ユーザ定義のマッピング

  このセクションは、ユーザが定義するドキュメントタイプをリンクするマッピングの詳細をまとめています。

  各マッピングごとに以下の情報が表示されます。
  - マッピングの名前

    マッピングの名前の後に、コネクタのソースドキュメントタイプとターゲットドキュメントタイプが括弧内に表示されます。
  - ソースドキュメントタイプの名前
  - ターゲットドキュメントタイプの名前
  - 各マッピングノードごとに以下の要素が表示されます。
    - ターゲット要素
    - マッピングスクリプト
    - ソース要素
    - マッピングの説明（必要に応じて）

  マッピング作成の詳細については、「ドキュメントタイプのマッピング [p. 87]」を参照してください。
- 関連するスケジューラとファイル

  このセクションの内容は以下の通りです。
  - シナリオのスケジューラ

    スケジューラの以下の情報が記載されます。
    - 名前
    - 周期
    - 関連するコネクタとドキュメントタイプ
  - マップテーブルファイル

- Basicスクリプトファイル
- 文字列ファイル

- シナリオ変更の履歴

  このセクションには、各シナリオごとに、全変更事項の情報が表示されます。これらの情報は、**［シナリオの履歴］**ウィンドウ（**［ファイル／シナリオの履歴］**メニュー）でシナリオ作成者が入力する情報に相当します。

シナリオ文書をHTMフォーマットで表示する

シナリオビルダには、現シナリオのHTMフォーマットの文書をインターネットブラウザ内に表示する機能があります。

シナリオ文書をHTMフォーマットで表示する

1. シナリオビルダでシナリオを開くか、または作成します。
2. **［ファイル／HTMLシナリオ文書を表示する］**を選択します。
3. インターネットブラウザが起動し、文書が表示されます。

注意:

［HTMLシナリオ文書を表示する］機能を使用するには、オペレーティングシステムで、HTMLファイル名の拡張子がインターネットブラウザ（例えばMicrosoft Internet ExplorerやNetscape Navigator）に関連付けられている必要があります。

シナリオ文書のプロパティ

シナリオ文書の作成には以下の要素を使用します。

- XSLプロセッサ

  Connect-It付属のプロセッサはXalanプロセッサです。
- HTMLフォーマットでの表示に使用するCSSスタイルシート
- SCNファイルをDBKフォーマットに変換するXSLスタイルシート
  - SCNファイルからDBKフォーマットへ
  - DBKファイルからHTMLフォーマットへ

SCNファイルのDBKフォーマットへの変換は、SCNファイルのXMLフォーマットへの変換を経由してから実行されます。

SCNファイルをHTMLフォーマットへ変換する場合、SCNファイルのXMLフォーマットへの変換を経由してから、HTMフォーマットへ変換されます。

シナリオ文書のフォーマット	変換順
DBK	1 SCN 2 XML 3 DBK
HTM	1 SCN 2 XML 3 DBK 4 HTM

シナリオ文書のプロパティを編集する

シナリオ文書を編集するには、［ファイル／シナリオ文書のプロパティ］を選択してウィザードを起動します。

XSLプロセッサ

ウィザードのこのページでは、SCNファイルをDBKフォーマットとHTMフォーマットに変換するために使用するXSLプロセッサを設定できます。

Connect-It付属のXalanプロセッサは、デフォルトのプロセッサです。

このファイルのパスは「［Connect-Itインストール先フォルダ］\datakit\doctrans\xalan\xalan.bat」です。

他のXSLプロセッサを使用することも可能です。例：MicrosoftのWebサイトで無料ダウンロード可能なMSXSLプロセッサなど。

XSLプロセッサの実行可能ファイル

このフィールドにXSLプロセッサの完全パスを指定します。デフォルトのパスは、[Connect-Itインストール先フォルダ]/datakit/doctrans/xalan/xalan.batです。

XSLプロセッサのパラメータ

このフィールドのデフォルトのパラメータは以下の通りです。

- %IN

  この変数は変換するSCNファイルの完全パスに相当します。
- %XSL

この変数は、DBKまたはHTMフォーマットへの変換に使用するXSLファイルの完全パスに相当します。

- %OUT

  この変数は、DBKまたはHTMの出力ファイルの完全パスに相当します。

HTMLシナリオ文書にCSSスタイルシートの使用を許可する

HTMフォーマットでシナリオ文書を表示する際にCSSスタイルシートを使用するためには、このオプションを選択します。

デフォルトで使用されるCSSスタイルシートのパスは、[Connect-Itインストール先フォルダ] \datakit \doctrans \css \default.cssです。

既製XSLスタイルシートのリスト

このページでは、以下の変換に使用するスタイルシートが表示されます。

- XMLファイルからDBKフォーマットへの変換
- DBKファイルからHTMフォーマットへの変換

［説明］と**［エイリアスまたはXSLスタイルシート］**の列の値を変更するには、値を直接クリックします。

スタイルシートのエイリアスは**［拡張子］**の列で入力する値です。スタイルシートの完全パスの前にセミコロンで区切ってエイリアスを入力すると（［エイリアス］；［完全パス］）、XSLプロセッサが最初に使用するスタイルシートはこのエイリアスのスタイルシートになります。

拡張子HTM用の値が、DBK;C: \Program Files \Peregrine \ConnectIt \datakit \doctrans \xsl \dbk2htm.xslである場合、XSLプロセッサがXMLフォーマットをまずDBKフォーマットへ変換してから、次にHTMフォーマットへ変換することを意味します。

ユーザ定義のXSLスタイルシートのリスト

このページでは、ファイルのフォーマットを変換するための新規XSLスタイルシートを指定できます。

ユーザ定義のXSLスタイルシートを指定する方法

1. をクリックします。
2. **［拡張子］**の列にファイルの拡張子を入力します。
3. 同じ行の**［説明］**の列をクリックして、説明を入力します。
4. 同じ行の**［エイリアスまたはXSLスタイルシート］**の列をクリックして、XSLスタイルシートのパスを入力します。

他のXSLスタイルシートのエイリアスを指定すると、最初のフォーマットと最終フォーマットの間の中間の変換を必要なだけ指定できるようになります。

シナリオ文書の作成

シナリオビルダでは、文書を以下のフォーマットで作成できます。

- HTMフォーマット
- DBKフォーマット

シナリオ文書の作成

1. シナリオビルダでシナリオを開くか、または作成します。
2. **［ファイル／シナリオ文書を作成する］**を選択します。
3. 文書ファイルに名前を付けます。
4. **［タイプ］**フィールドで文書のフォーマットを選択します。
5. **［保存］**をクリックします。

注意:

シナリオ文書は、デフォルトでシナリオのSCNファイルと同じフォルダ内に保存されます。

DBKフォーマットの文書

DBKフォーマットの文書を作成すると、DocBook DTDでXMLファイルが作成されます。

DocBook DTDの情報については、http://www.docbook.org のWebサイトを参照してください。

10 統合シナリオをプロダクションモードにする

シナリオをプロダクションモードにする場合、以下の操作により外部アプリケーションとシナリオの統合を自動化することができます。

- シナリオのソースコネクタがいつデータを処理するかを決めるスケジュールを作成する。
- シナリオに関連付けられたWindows32ビットのサービスを作成し、このサービスがConnect-Itサーバ上でバックグラウンドジョブとして実行されるようにする。
- Connect-Itコンソール経由で、処理中に発生する問題を管理する。

スケジュールの作成

スケジュールにより、ドキュメントを生成するコネクタがいつ起動するのかを指定できるようになります。スケジュールのないシナリオは不完全です。

スケジュールを作成するには、コネクタの生成用ドキュメントタイプをスケジューラに関連付けます。

スケジューラは、コネクタを起動するタイマの役割を果たしています。

- タイマはある時間帯内または時間帯外に、定期的にコネクタを起動します。または、
- ある日時に1回のみ（例えば2003年3月6日）起動します。

規則を作成すると、スケジューラを変更できます。例えばあるスケジューラがコネクタを毎日起動するとします。別の規則を作成すると、ある1日のスケジューラの機能を変更できます。

例 :

Desktop Discovery - Asset ManagementシナリオのDesktop Discoveryコネクタは、「Machine」というドキュメントを、毎日朝の9時から夜の10時まで5分おきに生成します。この時間帯以外は、Desktop Discoveryコネクタは1時間おきにドキュメントを生成します。これに別の規則を追加すると、以上のパラメータをある一定の期間用に変更することもできます。

スケジューラの編集

［シナリオ／スケジューラ］メニューを選択すると、スケジューラ編集ウィンドウが表示されます。

変更不可能なスケジューラが2つ提供されています。

- 「一回」スケジューラ

  このスケジューラは、ドキュメントの作成を一回起動します。生成用ドキュメントタイプが作成されると、デフォルトでこのスケジューラに関連付けられます。
- 「同期」スケジューラ

  このスケジューラは、0時から24時まで毎秒ドキュメントの生成を起動します。

図 10.1. スケジューラエディタ

アイコン	機能
	新規スケジューラの作成を起動します。
	選択したスケジューラを複製します。
	選択したスケジューラを削除します。

[実行日] フィールド

このフィールドでは、ドキュメントタイプが生成される日を指定します。

このフィールドには以下のオプションがあります。

- **[毎日]**

  年間を通じて毎日、例外なく実行します。

- **[日付指定]**

  [日] [月] [年] チェックボックスをオンにすると、1日または複数の日を選択できます。（例：2002年1月6日）

  第1、第2、第3、第4、最後から2番目、最終

［日］チェックボックスをオンにすると、曜日を、［月］［年］チェックボックスをオンにすると、月と年を指定できます。

例：毎月第1金曜日

［実行時間］フィールド

このフィールドには2つのオプションがあります。

- ［定期的］

  ［定期的］オプションを選択すると、1日のある期間を設定できます。**［期間内の実行間隔］**フィールドで間隔を指定します。

  例：朝4時から夜10時までの期間で、5分おきの実行。

  この期間外の実行の間隔は**［期間外の実行間隔］**で設定できます。

  テキストフィールドに値を入力します。手動入力には以下のシンタックスに従います。

  ```
  <開始時> -
  <終了時>, <開始時> - <終了時>...
  ```

  時間入力用のフォーマットは、例えばWindowsコントロールパネルでの地域オプションなどにより変化します。［AM|PM］オプションパラメータが指定されていないと、時間は24時間制で入力されているものと見なされます。
  例：「18」と入力すると、6:00 PMという値が確認後に自動的に表示されます。

  注意:

  グラフィカルエディタを使う場合は、30分単位で指定できます。直接に時間を数値で入力する場合は、1分単位で指定できます。

- ［項目リスト］

  時間のリストをセミコロンで区切って入力します。これにより、コネクタを選択した時間に起動することが可能になります。

  ```
  5:00AM;8:00PM;...
  ```

スケジューラの作成（例）

本節ではコネクタ起動用のスケジューラの作成方法を説明します。

- 毎日
- 朝8時から夜10時の間は10分おき、この期間外では1時間おきに起動

  これにドキュメントの生成用の以下の規則を追加します。

  - 毎月の第1日曜日
  - 朝6時から夜10時の間は10分おき、この期間外では30分おきに起動

スケジューラを作成するには、をクリックします。
スケジューラの詳細下に表示されるタブのフィールドに入力します。

毎日

［実行日］フィールドで［毎日］オプションを選択します。

朝8時から夜10時の間は10分おき、この期間外では1時間おきに起動

［実行時間］フィールドで［定期的］オプションを選択します。
朝8時から夜10時の期間を選択するには、以下の方法があります。

- テキストフィールドに直接入力します。（8:00 AM-10:00 PMまたは8-22）
- 24時間を表すグラフィカルエディタを使って入力します。

次に、［期間内の実行間隔］と［期間外の実行間隔］フィールドに「10分」と「1時間」を入力します。このフィールドは、選択した期間内と期間外での起動の頻度を設定します。
以上の手順に従うと、タブの内容は以下のようになります。

このタブは、［実行日］フィールドに入力されたデータにより自動的に名前が付けられます。
［作成］をクリックし、スケジューラを確定します。

規則の作成

規則を作成するには、スケジューラエディタのタブ上で右クリックします。
ショートカットメニューで［規則の追加］オプションを選択します。
新規のタブが表示されます。スケジュールに追加する規則を入力します。

この例（毎月の第1日曜日、朝6時から夜10時の間は10分おき、この期間外では30分おきに起動）では、タブの内容は以下のようになります。

［作成］をクリックして規則を確定します。

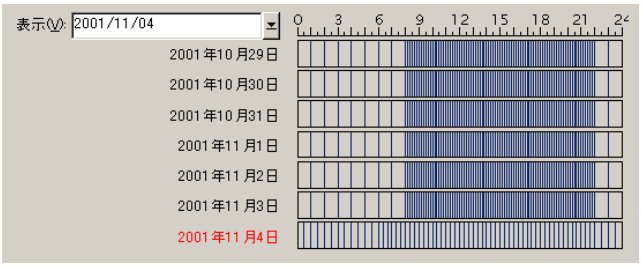
規則の削除

規則を削除するには、規則のタブ内で右クリックします。ショートカットメニューで［規則の削除］オプションを選択します。

プレビュー

［プレビュー］タブでは、現在の週のスケジューラの予定が表示されます。

［表示］フィールドの日付を選択すると、カレンダは選択した日付の週のスケジューラを表示します。



スケジューラの変更

スケジューラを変更するには、

1 リストからスケジューラを選択します。
2 タブ内でパラメータを変更します。
3 ［変更］をクリックして変更事項を確定します。

スケジューラの削除

スケジューラを削除するには、

- リストからスケジューラを選択します。
- ［Delete］キーを押すか、または をクリックします。

スケジュールの編集

スケジュールを編集するには、［シナリオ／スケジュール］メニューを使用します。

シナリオのデフォルトのスケジュールは、生成用ドキュメントタイプを「一回」スケジューラに関連付けています。

生成用ドキュメントタイプをスケジューラに関連付けるには、

- 生成用ドキュメントタイプをクリックし、選択したスケジューラにドラッグします。

  または

- 生成用ドキュメントタイプを選択し、編集ウィンドウの右側にある矢印を使って移動させます。

スケジューラは、編集ウィンドウにアルファベット順に表示されます。スケジューラを、シナリオの生成用ドキュメントタイプに関連付けます。

図 10.2. スケジュールの編集ウィンドウ

スケジュール内の生成用ドキュメントタイプの順番

生成用ドキュメントタイプを別のスケジュールへ移動させる	
⏫	前のスケジュールへ移動する
⏬	後のスケジュールへ移動する
同じスケジュール内でドキュメントタイプを移動させる	
🔼	上に1行移動する
🔽	下に1行移動する

データの処理方法は以下の2つの要因に左右されます。

- スケジュール内の生成用ドキュメントタイプの順番
- 同じ生成用ドキュメントタイプ内のマッピングの順番

ドキュメントタイプが並べられている順番により、生成の順番が決定されます。

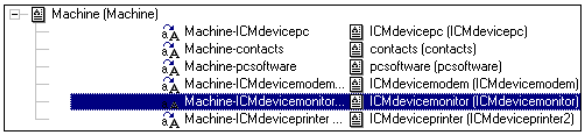
Connect-Itメインウィンドウの［**マッピング**］タブ内のマッピングの順番も重要です。同じソースドキュメントタイプに複数のマッピングがある場合、［マッピング］タブ内のマッピングの順番により、Connect-Itがどのマッピングを最初に実行するかが決まります。

マッピングを並べ替えるには、

- シナリオ図内でマッピングボックスを選択します。
- ［**マッピング**］タブを選択します。
- リスト内で生成用ドキュメントタイプを選択します。
- 矢印をクリックして、リスト内でドキュメントタイプを移動させます。

例：Desktop Discovery - ServiceCenterシナリオでは、**Machine**（Machine）ドキュメントタイプに6つのマッピングがあります。

図 10.3. マッピングの順番



アイコン	機能
	マッピングの作成を起動します。
	選択したマッピングを編集できるようになります。
	選択したマッピングを削除します。
	マッピングを上へ1行移動させます。
	マッピングを下へ1行移動させます。

シナリオを起動する際、Connect-Itは同一のスケジューラ内では以下の順番で優先度を決定します。

- 生成用ドキュメントタイプの順番
- 同じ生成用ドキュメントタイプ用のマッピングの順番

ポインタのステータス

多くの場合、ポインタのステータスは特定の時間を指しています。ポインタに提供されるこの情報は非常に大切です。これにより、外部アプリケーションのデータ処理の進み具合を確認できます。

ポインタのタイプは関連する外部アプリケーションにより変わります。

- Asset Managementコネクタ

  ポインタは、シナリオ内の各マッピングの最終起動日時に当たります。このポインタの値よりも、最終変更日（［dtLastModif］フィールド）が後のレコード全てが処理されます。

  注意:

  Connect-Itがインストールされているコンピュータと、AssetCenterサーバ間の時間差は自動的に補正されます。

- NTセキュリティコネクタ

  スケジューラが起動するたびに、コネクタは補足情報だけではなく、検索するドメインの全情報を取得します。

  注意:

  新規スケジューラを作成し、そのスケジューラをNTドメインに割り当てる場合、スケジュールのポインタは表示されません。

- Desktop Discoveryコネクタ

  使用されるポインタは、既に処理された「.fsf」ファイルの最も最新の作成日です。

- Network Discoveryコネクタ

  使用されるポインタはイベントの番号です（EVENTドキュメントタイプの場合）。

- ServiceCenterコネクタ

  このコネクタのポインタは、ソースアプリケーションから来るデータの性質により変化します。

  - イベントの場合、ポインタは連続番号（evsysseq）に当たります。

    ポインタは、シナリオの各マッピングの最終の起動時に処理されたイベントの連続番号に当たります。次回の起動時には、このポインタ値よりも高い連続番号のイベント全てが処理されます。

  - テーブルの場合、ポインタの値はマッピングの最終起動日です。次回の起動時には、このポインタよりも、最終変更日（［sysmodtime］フィールド）が後のレコード全てが処理されます。

注意:

ServiceCenterコネクタの設定により、Connect-Itがインストールされているコンピュータと、ServiceCenterサーバ間の時間差は以下の方法で処理されます。

- 時間差は自動的に補正されます。
- ユーザの入力した時間差に応じて補正されます。

各ポインタのステータスをダブルクリックすると、「.fsf」ファイルの別の日付、またはNetwork Discoveryの別のイベント番号を強制できます。

例

Network Discoveryは数万個のイベントを記憶します。ポインタのステータス（スケジュールの最初の起動前の値は「0」です）を20000にすると、番号が20000より大きなイベントのみがシナリオで処理されます。

「.fsf」ファイルのポイント

例 :

InfraTools Desktop Discoveryコネクタの第1回目の使用時に、次の2つのファイルがあるとします。

- 1月1日の「paris.fsf」ファイル
- 1月15日の「rome.fsf」ファイル

コネクタを次回に起動する際、1月15日という日付が基準日（ポインタ）になります。このため、この日付より前に作成されたファイルは**処理されません**。1月15日より前の「.fsf」ファイルの処理をシナリオ内で強制するには、スケジュールの編集ウィンドウで、ポインタのステータスを直接変更する必要があります。

シナリオのプロダクションモードでは、［フォルダに保存する］オプションがInfraTools Desktop Discoveryコネクタの設定で選択されていると、ファイルの一番最近の日付が、InfraTools Desktop Discoveryコネクタの基準日（ポインタ）になります。

スケジュールの編集ウィンドウ内でのポインタを更新する

シナリオを起動する時にスケジュールの編集ウィンドウが開いていると、ポインタのステータスは更新されません。スケジュールの編集ウィンドウのポインタのステータスを更新するには、一度このウィンドウを閉じてから新たに開く必要があります。

スケジュールの起動

スケジュールを起動するには、シナリオを起動する必要があります。シナリオを起動するには、

- **［シナリオ／全スケジューラを起動する］**オプションを選択します。

  または

- をクリックします。

スケジュールの停止

スケジュールを停止するには、シナリオを停止する必要があります。シナリオを停止するには、

- **［シナリオ／停止］**オプションを選択します。または、
- をクリックします。

Connect-Itサービスの作成

Connect-Itでは、Windows上のサービスをシナリオに関連付けることができます。このサービスにより、Connect-Itサーバはバックグラウンドジョブとしてデータ処理を実行できます。データ処理は、シナリオに関連付けられたスケジュールに応じて起動されます。

警告:

シナリオ内の1つのコネクタがODBC接続を使用している場合、この接続はシステムデータソース（システムDNS）を使用しなければなりません。この接続がユーザデータソース（ユーザDNS）を使用していると、接続はシナリオに関連付けられたサービスの管理下に置かれなくなります。

Windows下のサービスでは、バックグラウンドジョブとして実行されるアプリケーションを使用できます。Connect-Itでは、シナリオと同じ数だけのサービスを作成、起動できます。

注意:

Windows 32ビットでは、環境変数（例えば検索パスなど）を変更しても、コンピュータを再起動しない限りこの変更事項はサービスに適用されません。

重要項目:

コネクタが他のコンピュータのフォルダやファイルを指定する場合は、マップされたディスク名は使わずに、遠隔のコンピュータ名を直接指定します。例えばディスク「z」が「\\FSFStore」にマップされている場合、

```
\\FSFStore\Scan
```

が正しい指定方法で、

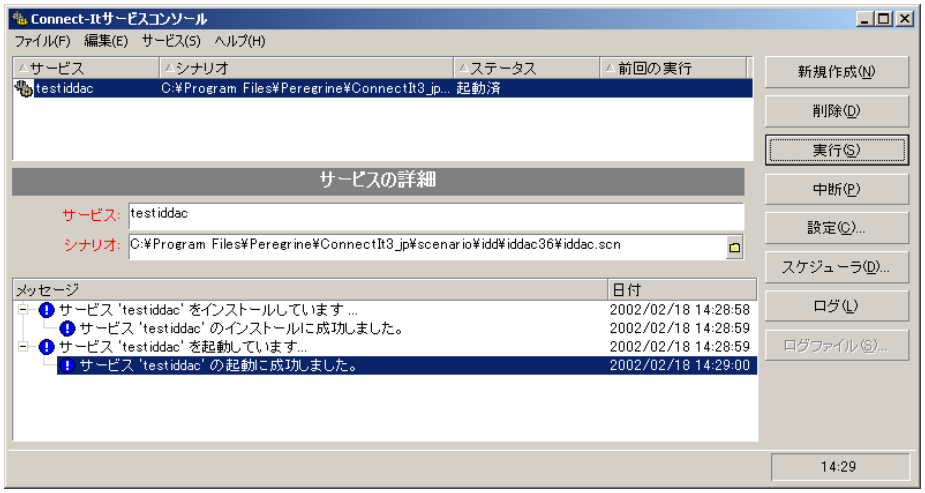
```
Z:\Scan
```

は間違いです。

サービスコンソール

サービスコンソールはConnect-Itアプリケーションのコンポーネントです。コンソールのグラフィカルインタフェースでは、シナリオを管理し、シナリオに関連付けるサービスを作成することができます。

図 10.4. サービスコンソール



コンソールを起動するには、［サービスコンソール］を、Windowsの［スタート］メニュー内の［Peregrine\Connect-It］プログラムグループから選択します。または、Connect-Itインストール先フォルダの「bin32」フォルダ内の、「console.exe」実行可能ファイルを起動することもできます。

メニュー

サービスコンソールには4つのメニューがあります。

［ファイル］メニュー

コマンド	機能
終了	Connect-Itコンソールを終了します。

［編集］メニュー

コマンド	機能
切り取り、コピー、貼り付け	コンソール用の標準の編集機能

［サービス］メニュー

コマンド	機能
実行	Connect-Itサービスを起動します。
中断	Connect-Itサービスを停止します。
設定	**［コネクタの設定］**ウィザードを起動し、シナリオのコネクタの設定を可能にします。
スケジューラ	サービスに関連付けられたシナリオ内で使用されているスケジューラを変更できるようにします。
ログ	シナリオビルダを起動し、**［Connect-Itログ］**と**［ドキュメントログ］**タブを表示します。
ログファイル	作成されたConnect-Itの各サービスに付属するログファイルを表示します。

［ヘルプ］メニュー

コマンド	機能
バージョン情報...	［バージョン情報］ボックスが表示されます。

WindowsでConnect-Itサービスを作成する

Connect-Itサービスを作成する

1. サービスコンソールを起動します。
2. **［新規作成］**をクリックします。
3. **［サービス］**フィールドにサービスの名前を入力します。
4. をクリックして、シナリオの名前とパスを指定します。
5. **［作成］**をクリックします。

   コンソールの上部枠に作成されたサービスが表示されます。

サービスの接続プロパティ

Windowsのサービスが正常に機能するためには、サービスの接続プロパティを変更しなければならない場合もあります。

例

- サービスに関連するシナリオのソースコネクタが、ネットワークドライブ上にあるフォルダを読み取ることになっているが、サービスのローカルアカウントはこのネットワークドライブにアクセスできない。
- シナリオのEメールコネクタはある特定のアカウントでしか機能しない。

上記の2つの例では、サービスの接続プロパティ内で特定のアカウントを指定しなければなりません。

Windows XP上サービスの接続プロパティを変更する

1. Windows XPを起動します。
2. **［スタート／パラメータ／コントロールパネル／管理ツール／サービス］**を選択します。
3. リスト内のサービスをダブルクリックします。
4. 表示されるダイアログボックスの**［接続］**タブを選択します。
5. **［アカウント］**オプションを選択します。
6. サービスに関連したシナリオを正しく使用できるように、適切な値を**［アカウント］**と**［パスワード］**フィールドに入力します。
7. **［適用］**をクリックします。

シナリオのパス

［シナリオ］フィールドにシナリオのパスを入力する場合は、以下の規則に従います。

1. シナリオはConnect-Itサーバに保存されている。

シナリオが、ネットワークドライブのフォルダ内にある場合は、ネットワークドライブの文字を［シナリオ］フィールド内に入力しないでください。

例：シナリオ「scenario \myscenario.scn」は、ネットーワークドライブR:に関連付けられたフォルダ「C: \Program Files \ConnectIt」内に位置します。この場合、［シナリオ］フィールドには、「C: \Program Files \ConnectIt \scenario \myscenario.scn」を指定し、R: \scenario \myscenario.scnとは入力しないでください。これは、サービスがローカルシステムアカウントまたは特定のアカウントを使用してるためです。これらのアカウントでは、C: \Program Files \ConnectItフォルダは、ネットワークドライブR:に関連付けられていません。

2 シナリオはネットワーク上にインストールされている。

サービスが、［シナリオ］フィールド内に指定されているネットワークパスを使用できるかどうか、確認してください。

サービスがネットワークパスを使用できるようにするには、このネットワークパスを使用できるユーザ名とパスワードに、サービスを関連付けます。

UNIXでConnect-Itサービスを作成する

UNIXでConnect-Itサービス（デーモン）を作成するには、

1 クライアントのルートディレクトリにある「conitsvc.ini」ファイルを編集します。

2 [Service] セクションを作成し、以下の行を追加します。

```
[サービス名]=[シナリオの完全パス]
```

```
[Service]
indsc=/export/home/jean/test/ConnectIt/scenario/ind/indsc.scn
```

例

このUNIXデーモンを起動するには、以下のコマンドラインを実行します。

```
conitsvc -svc indsc &
```

Connect-Itサービスの起動または停止

Connect-Itサービスを起動するには、

- Connect-Itコンソールを起動します。
- Connect-Itコンソールのメイン枠内で、サービスを選択します。
- ［実行］をクリックします。

注意:

サービスを起動するには、シナリオを事前に設定する必要があります。

Connect-Itサービスを停止するには、

- Connect-Itコンソールを起動します。
- Connect-Itコンソールのメイン枠内で、サービスを選択します。
- **［中断］**をクリックします。

Connect-Itサービスの削除

Connect-Itサービスを削除するには、

- Connect-Itコンソールを起動します。
- Connect-Itコンソールのメイン枠内で、サービスを選択します。
- **［削除］**をクリックします。

コマンドライン

コマンドラインからサービスを起動するには、

1. Connect-Itのインストール先フォルダの、**「bin32」**サブフォルダに移動します。
   例：C:\Peregrine\ConnectIt\bin32\
2. コマンドラインで以下の行を実行します。
   NetStart <サービス名>

例：

Asset Management-ServiceCenterサービスを起動する場合のコマンドラインは以下の通りです。

NetStart Asset Management-ServiceCenter

シナリオのトラッキングをサービスコンソールで管理する

サービスコンソールを使用すると、シナリオのトラッキング管理用の以下の操作を実行できます。

- コネクタの設定を変更する
- シナリオのスケジューラを編集する
- Connect-Itログを参照する
- サービスに関連するLOG（ログ）ファイルを参照する

コネクタの設定の変更

作成したサービスに関連するシナリオのコネクタの設定を変更するには、

1 シナリオに対応するサービスを選択します。
2 **［設定］**をクリックします。
3 **［コネクタの設定］**ウィザードの各ページに変更事項を入力します。

最後のコネクタの設定を終了すると、シナリオビルダは自動的に終了します。

スケジューラの編集

作成したサービスに関連するシナリオのスケジューラを編集するには、

1 シナリオに対応するサービスを選択します。
2 **［スケジューラ］**をクリックします。
3 Connect-Itが起動し、スケジューラエディタのウィンドウが表示されます。

コネクタの編集の詳細については、本章の「スケジュールの作成 [p. 163]」の節を参照してください。

Connect-Itログの参照

シナリオビルダでは2つのログを参照できます。

- Connect-Itログ

  このログは、シナリオの起動時にConnect-Itで実行された全アクション（シナリオの非シリアル化、外部アプリケーションへのコネクタの接続、など）を記述します。
- ドキュメントログ

  このログでは、シナリオのコンポーネントにより生成された、または取り込まれたドキュメントの詳細を確認できます。このログではまた、ドキュメントの処理中に発生する問題の原因を突き止めることができます。

シナリオビルダのログ用のタブを表示するには、

1 シナリオに対応するサービスを選択します。
2 **［ログ］**をクリックします。

シナリオビルダが、シナリオ図とConnect-Itログに対応するタブと共に起動します。

サービスのログファイルの参照

Connect-Itサービスが起動するたびに、ログファイルが使用可能になります。

ファイルにはサービス名がつけられています（例：「test.log」）。ログファイルを読むには、

1 シナリオに対応するサービスを選択します。

2 **［ログファイル］**をクリックします。

シナリオのログファイルは、オペレーティングシステムで「.log」ファイルに関連付けられたアプリケーション内で開きます。

シナリオの性能を最適化する

本節では、シナリオのコンポーネントがドキュメント処理に費やす時間を短縮することによって、シナリオの性能を最適化する方法を説明します。

実行結果を使ってドキュメント処理の時間を評価する

シナリオビルダのグラフィカルインタフェースの［Connect-Itログ］、またはログファイルでは、以下の内容に関する実行結果を取得できます。

- ソースコネクタがドキュメントの生成に費やした時間
- マッピングボックスがドキュメントの変換に費やした時間
- ターゲットコネクタがドキュメントの取り込みに費やした時間
- 拒否されたドキュメントの数

例：Asset Managementコネクタの実行結果は、製品のテーブルのレコードに対応するドキュメントが処理されなかったことを示します。

コンポーネントのデータ処理の実行結果を取得する

1 シナリオを開きます。

2 シナリオの全コネクタを開きます（**［Ctrl+F4］**キー）。

3 シナリオのソースコネクタを選択します。

例：idd\iddac36\iddac.scnシナリオのInfraTools Desktop Discoveryコネクタ

4 ▸をクリックします（**［F5］**キー）。

5 ソースコネクタ、ターゲットコネクタ、またはマッピングボックス上にマウスのポインタを置きます。

［実行結果］欄を含む状況依存ウィンドウが表示されます。

注意:

これらのデータは、シナリオ図の下にある［Connect-Itログ］タブ内にも表示されます。

実行結果の分析

Connect-Itログ内に表示される以下のデータは、Action Request Systemソースコネクタ、マッピングボックス、LDAPターゲットコネクタを含むシナリオの実行結果です。

```
コネクタ 'Action Request System (fdcitsrv01)' の実行結果 (セッション: 26分  57.310秒 / API: 26分  04.550秒)
取り込まれたドキュメント: 7143
拒否されたドキュメント: 1
挿入されたレコード: 1199
更新されたレコード: 5943
コネクタ 'LDAP (mail-sd.peregrine.com)' の実行結果 (セッション: 47.628秒 / API: 35.765秒)
生成されたドキュメント: 7143
コネクタ 'Mapping (Basic engine)' の実行結果 (セッション: 01.404秒)
評価されたBasic script: 14286
取り込まれたドキュメント: 7143
生成されたドキュメント: 7143
```

作成された実行結果は以下のように分析されます。

- Action Request Systemコネクタによるドキュメントの処理 = 27分、または1秒につき4.4ドキュメント

  処理時間のうち、Action Request Systemに関連するAPIにかかった時間が26分で、Connect-Itでの処理にかかった時間は1分です。

  APIは、ネットワークの応答時間、「commit」の実行、データベースの保全性の管理などを含みます。
- LDAPコネクタによるドキュメントの処理=48秒、または1秒につき150ドキュメント

  処理時間のうち、LDAP APIでかかった時間が36秒で、Connect-Itでの処理にかかった時間は12秒です。
- マッピングボックスによるドキュメントの処理：2秒、または1秒につき5100ドキュメント

  この時間は、Connect-Itでのドキュメント処理の時間に相当します。

ドキュメント処理の速度の例

ドキュメント処理の速度の例は以下の通りです。

- データベースコネクタ
  1500から2000ドキュメント／秒
- ServiceCenterコネクタ
  450ドキュメント／秒
- AssetCenterコネクタ
  400ドキュメント／秒

ドキュメントの生成を改善する

コネクタのドキュメント生成を改善するには、ソースアプリケーションから来るデータをより迅速に取得することが必要です。

本節では、ドキュメント生成にかかる時間を短縮する方法を説明します。

ソースデータサーバの最適化

ドキュメント生成の性能は、ソースデータサーバの使用方法とサーバの接続方法に左右されます。

Connect-Itが実行するデータの変換やデータの生成処理はごくわずかです。

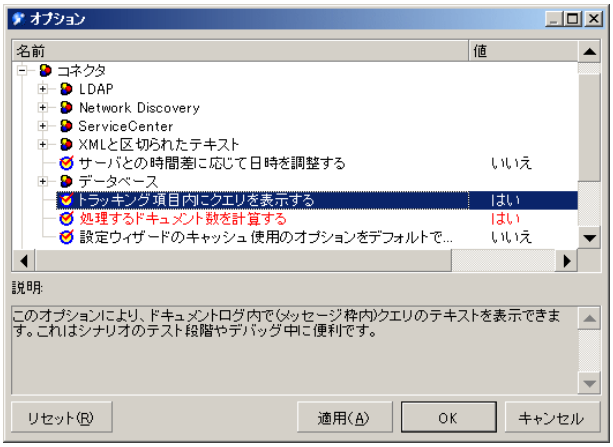
結果を改善するためには、以下の内容を確認する必要があります。

- サーバのデータソースの読み込み
  読み込みの機能はサーバの技術的な性能（プロセッサ、使用可能なメモリなど）に左右されます。
- ネットワーク接続（WAN、LAN)

コネクタが実行するクエリの検証

ソースデータベースへ伝達されるクエリをトラッキングするには、

1. シナリオビルダを起動します。
2. **［編集／オプション］**を選択します。
3. **［コネクタ］**項目を選択して全レベル表示します。
4. **［トラッキング項目内にクエリを表示する］**オプションで、**［はい］**を選択します。
5. **［OK］**をクリックします。



このオプションが選択されると、シナリオのソースコネクタが実行するクエリに対応するメッセージが、ドキュメントログ内に表示されます。

例

```
SELECT AcctCode,AssetTag,BarCode,dDispos,DisposProfit,dtListPriceCv,Field2,FullName FROM amAsset
```

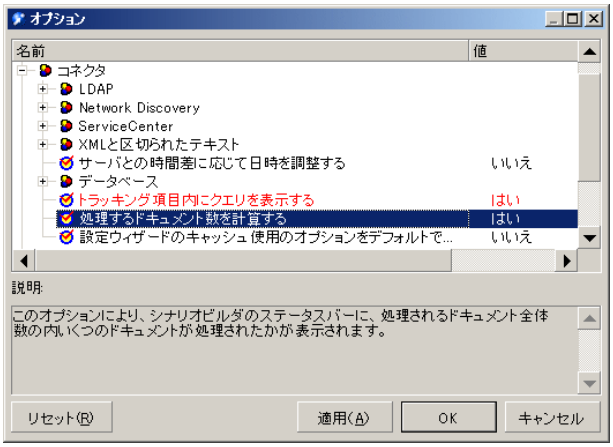
この情報により、以下の操作が可能になります。

- 指定したドキュメントタイプと生成用ルールに従って、クエリが生成されているかどうかを確認する
- 別のツールを使ってクエリを再起動し、クエリ実行に必要な時間を監視する

生成するドキュメント数の計算

ソースコネクタが生成するドキュメントの数を計算するには、

1. シナリオビルダを起動します。
2. **［編集／オプション］**を選択します。
3. **［コネクタ］**項目を選択して全レベル表示します。
4. **［処理するドキュメント数を計算する］**オプションで、**［はい］**を選択します。
5. **［OK］**をクリックします。



このオプションを使用すると、ソースコネクタが処理するドキュメントの数を表示する進行状況バーが、シナリオビルダ内に表示されます。

図 10.5. ドキュメントの処理 - 進行状況バー

Asset Managementコネクタで、このオプションはSELECT COUNT型のクエリを起動します。

```
SELECT COUNT(AcctCode) FROM amAsset
```

例

警告

このクエリの実行には時間がかかるため、シナリオのプロダクションモードでは、このオプションをオフにしてください。

コネクタのキャッシュオプションの使用

データベース型のコネクタでは、生成用ドキュメントタイプ（メタデータ）の構造にキャッシュを使用することが推奨されています。

キャッシュを使用すると、生成用ドキュメントタイプ（メタデータ）の記述の読み込みが、ローカルコンピュータで実行されるため、コネクタがより速く開くようになります。

コネクタのキャッシュ使用の詳細については、マニュアル『**コネクタ**』の「**コネクタの設定**」の章の「**キャッシュを設定する**」の節を参照してください。

キャッシュを使用しない場合、コネクタは、コネクタを開く度にソースデータベースの記述に対応するデータ全体を取得します。

データベース型コネクタでは、データベース記述は次のデータに当たります。

- テーブルのリスト
- フィールドのリスト
- 書式のリスト
- インデックスのリスト
- リンクのリスト
- 結合のリスト
- その他

上記のデータが多数であるほど、コネクタを開く際に時間がかかります。

注意:

コネクタを開く時に使用可能な帯域幅が低い場合、かかる時間は長くなります。これはネットワークの接続（WAN、LAN）の性能に左右されます。

警告

シナリオのプロダクションモードでは、キャッシュの使用が推奨されています。

しかし、ソースデータベースの記述内容が変更した場合は、コネクタのキャッシュの同期をとらなければなりません。

コネクタのキャッシュの同期をとるには、

1 シナリオビルダを起動します。
2 シナリオを開きます。
3 キャッシュの同期をとる必要があるコネクタを選択します。
4 コネクタを開きます（［F4］キー）。
5 **［ツール／キャッシュ／キャッシュの同期をとる］**を選択します。

キャッシュの同期をとった後、ソースデータベース記述の新規データに存在しない要素が、マッピングに含まれていないかどうか確認します。

自動再接続

データソースとして遠隔のサーバを使用するコネクタでは、サーバへの自動再接続のオプションを選択する必要があります。

自動再接続のオプションは、ネットワークでの待ち時間（データパケットの伝達）に多大な影響を与えます。

自動再接続のオプションについては、マニュアル『**コネクタ**』の**「コネクタの設定」**の章の**「再接続のパラメータを設定する」**の節を参照してください。

再接続に関する情報は、Connect-Itログ内に表示されます。

Action Request Systemコネクタの再接続試行の例

```
サーバへ接続中...
ダイナミックライブラリ 'arapi50.dll'の使用。
(ARS ERROR 90) ARシステムサーバーへネットワーク接続できません。 fcitsrv0
1 : RPC: Name to address translation failed - No such hostname
サーバへの次回の再接続は4秒後に試行されます。
再接続を試行しています...
ダイナミックライブラリ 'arapi50.dll' の使用。
(ARS ERROR 90) ARシステムサーバーへネットワーク接続できません。fcitsrv0
1 : RPC: Name to address translation failed - No such hostname
サーバへの次回の再接続は8秒後に試行されます。
再接続を試行しています...
ダイナミックライブラリ 'arapi50.dll' の使用。
(ARS ERROR 90) ARシステムサーバーへネットワーク接続できません。 fcitsrv0
1 : RPC: Name to address translation failed - No such hostname
サーバへの次回の再接続は16秒後に試行されます。
```

ネットワークへの接続が不安定な場合、待ち時間は長くなります。

一定期間以上経った接続を切断する**ルータ／ファイヤウォール／サーバ**型の接続設定には、注意が必要です。この種のパラメータについては、ローカルネットワークの担当部署で確認してください。

各セッション毎の再接続（Asset Managementコネクタ用）

Asset Managementコネクタ専用の設定オプションを使用すると、シナリオの各セッションの終了時に接続を切断し、新規セッションの開始時のみに接続を再開するように設定できます。

Connect-Itサーバのリソースが保存されるほど、サーバへより速く再接続されるようになります。

このオプションの詳細については、マニュアル『**コネクタ**』の「**Peregrine Systems コネクタ**」の章の「**Asset Managementコネクタ**」の節、「**Asset Managementコネクタの設定**」を参照してください。

警告

シナリオの実行頻度が高い場合（5分おきなど）は、このオプションを無効にする必要があります。

スケジュールを使ったシナリオ実行の頻度設定については、「統合シナリオをプロダクションモードにする [p. 163]」章の「the section called “ スケジュールの作成 ” [p. 163]」の節を参照してください。

WHERE句の使用の最適化

データベース型コネクタの生成用ルールでは、WHERE句を使用できます。

WHERE句作成に関する詳細は、マニュアル『**コネクタ**』の「**コネクタのルール（ディレクティブ）**」の章、「**生成用ルール**」の節の「**WHERE句とORDER BY句**」を参照してください。

WHERE句の使用を最適化するには、句の適用先のフィールドがインデック化されている必要があります。WHERE句がインデックス化されていないフィールドに適用されると、句を実行する際にテーブルの全レコードが検索されることになります。

ポインタの管理

シナリオが処理するデータ量を少なくするために、ポインタを使用することも可能です。

ポインタがレコードの最終変更日である場合、コネクタは最後の起動以降に作成または更新されたレコードのみを処理します。

例：ソースコネクタは、［従業員］テーブルのレコードのうち、企業の新入社員に対応するレコードのみを処理します。

ポインタの使用方法については、マニュアル『**コネクタ**』の「**コネクタの設定**」章の「**スケジュールのポインタを設定する**」の節を参照してください。

一般的に、ポインタのステータスは特定の日時を指します。日時は、シナリオのスケジュール用ウィンドウ（**［シナリオ／スケジュール］**メニュー）で変更できます。

シナリオのポインタがインデックス化されていないフィールドを使用すると、データの処理にかかる時間は長くなります。処理時間を短縮するには、ポインタとして使用するフィールドを、インデックス化することが推奨されています。

ドキュメントの取り込みを改善する

コネクタのドキュメント取り込みを改善するには、ターゲットアプリケーションでデータをより迅速に処理することが必要です。

本節では、ドキュメントの取り込みにかかる時間を短縮する方法を説明します。

識別キーの選択

整合性チェックは旧データと新規データの比較に当たります。データベース型のコネクタ（Asset Management）では、コネクタが作成または更新するテーブルレコードを一意な方法で識別するためのフィールドを定義することが、整合性チェックに相当します。

識別キーとしてどのフィールドを選択するかは、ターゲットコネクタのドキュメントの取り込みに大きく影響します。

ドキュメントの取り込みは、以下の2つのアクションが実行されることを意味します。

1 識別キーとして選択されたフィールドを使ってクエリを送信し、ターゲットアプリケーションにレコードが存在するかどうかを確認します。

2 レコードの挿入または更新のアクションが実行されます。

識別キーとして選択されるフィールドがインデックス化されていない場合は、ドキュメントの取り込み速度は遅くなります。例：memoやFeatParamフィールドが、Asset Managementコネクタ用の識別キーとして選択されていると、ドキュメントの取り込みの効率は悪化します。

トランザクションの管理

データベース型コネクタは、1ドキュメントを取り込むたびに、デフォルトで「コミット」を実行します。

使用する帯域幅が低い場合、ドキュメントのグループごとにコミットを実行することも可能です。

何個のドキュメントごとにコミットするかを指定するには、

1 シナリオビルダを起動します。

2 シナリオを開きます。

3 トランザクションのパラメータを変更する必要があるコネクタを選択します。

4 コネクタの設定ウィザードを起動します（［F2］キー）。

注意：コネクタの高度な設定が有効になっている（ツールバーの アイコンがオンになっている）かどうか確認します。

5 **［次へ］**を数回押して、**［トランザクションを管理する］**ページを開きます。

6 **［ドキュメントグループごとにコミットする］**オプションを選択します。

7 ドキュメント数を記入して、いくつのドキュメントごとにコミットするかを指定します。

莫大な量のドキュメントを取り込む場合は、多数のドキュメントグループごとのコミットが推奨されています。

シナリオ「ldap\ac\complete.scn」の性能向上の例

1 各ドキュメントごとのコミット（デフォルトのオプション）

性能：6000ドキュメント用に4分（25ドキュメント／秒）

2 500ドキュメントのグループごとのコミット

性能：6000ドキュメント用に3分（33ドキュメント／秒）

同期処理と非同期処理 - ServiceCenterコネクタ

ServiceCenterコネクタが取り込む各ドキュメントは、ターゲットアプリケーションへ送信されるクエリに相当します。

ドキュメントの取り込み速度を上げるには、以下のどちらかを選択する必要があります。

- データの同期処理

  取り込まれる各ドキュメントは、ターゲットアプリケーションが前のドキュメントを処理した後に送信されます。
- データの非同期処理

  取り込まれる各ドキュメントは、ターゲットアプリケーションが前のドキュメントを処理していなくても送信されます。

Connect-Itエンジンを使ってドキュメント処理を改善する

本節では、Connect-Itエンジンを使ってドキュメント処理の時間を短縮する方法を説明します。

ドキュメントログの設定

Connect-Itのドキュメントログでは、以下の要素を指定できます。

- ドキュメントログ内に表示するエラーの種類（**［フィルタ］**フィールド）
- Connect-Itサーバのメモリに保存されるトラッキング項目の最大数
- ドキュメントログのメッセージを保存するテキストファイル

データ処理にかかる時間を短縮するには、

1. シナリオビルダを起動します。
2. **［ログ／ドキュメントログを設定する］**を選択します。
3. **［フィルタ］**フィールドで1項目を選択し、ドキュメントログ内に保存するエラーの種類を限定します。

   例：拒否
4. 保存するトラッキング項目の数を制限します。

   トラッキング項目の最大数は5000です。

［ファイルを使用］オプションを使用しない場合、Connect-Itエンジンのリソースを節約できます。

非グラフィカルモードの使用

シナリオの進行中にシナリオビルダが表示されていると、コマンドが選択されるたび（更新、ドキュメントログの閲覧、新規オプションの入力など）にConnect-Itエンジンは減速します。

スケジュールの規則

Connect-Itのスケジュールは、シナリオの生成用ドキュメントタイプに1つまたは複数のスケジューラを関連付けることによって作成されます。例：あるスケジューラはAsset Managementコネクタを1時間ごとに起動します。起動するごとに、コネクタはAsset Managementアプリケーションのレコードに対応するドキュメントを生成します。

コネクタが起動するたびに、Connect-Itは処理するデータの量を計算し、データ全体が処理されるまでConnect-Itは停止しません。処理するデータ量に応じて、シナリオのスケジューラを調整する必要があります。大規模なデータベースのマイグレーションの場合は、稼動率が低い時期（夜間のスケジュールなど）にソースコネクタの起動をスケジュールすることが推奨されています。少量のデータを処理する統合シナリオの場合は、既製の**「同期」**スケジューラの使用が推奨されています(ソースコネクタが毎秒起動）。

スケジュールの詳細については、本章の「スケジュールの作成 [p. 163]」の節を参照してください。

シナリオの並行起動

1つのシナリオが多数のデータを移行する場合、シナリオを複数のシナリオに分割し、各シナリオを別々のサービス（Windows）またはデーモン（UNIX）で起動させる方法が推奨されています。

サービスとデーモンの詳細については、本章の「Connect-Itサービスの作成 [p. 173]」の節を参照してください。

例

データベースに記録された従業員のリストを、別のデータベースにシナリオがインポートするとします。

移行にかかる時間を縮小するには、以下の2つのシナリオを作成します。

- 1番目のシナリオは、姓のイニシャルがAからJである従業員のリストを移行します。
- 2番目のシナリオは、姓のイニシャルがKからZである従業員のリストを移行します。

従業員を選択するには、2つのシナリオのソースコネクタで生成用ルール（WHERE句）を作成します。

WHERE句作成に関する詳細は、マニュアル『**コネクタ**』の「**コネクタのルール（ディレクティブ）**」の章、「**生成用ルール**」の節の「**WHERE句とORDER BY句**」を参照してください。

Asset Managementコネクタを使用するシナリオの性能を向上させる

本節では、Asset Managementコネクタを使うシナリオで、ドキュメントの処理にかかる時間を短縮する方法をいくつか紹介します。

dtLastModifフィールドのインデックス化

Asset Managementコネクタを使用するシナリオでは、シナリオに関連するAssetCenterの全テーブルの［**dtLastModif**］フィールドにインデックスを追加しなければなりません。Connect-Itはこのフィールドを一貫して使用して、最終セッション後に作成または変更されたレコードを確認します。

資産のテーブル（amAsset）、製品のテーブル（amProduct）、従業員と部署のテーブル（amEmplDept）でインデックスを作成するには、以下のコマンドを実行します。

```
CREATE INDEX Ast_dtLastModif ON amAsset (dtLastModif)
GO
CREATE INDEX Prod_dtLastModif ON amProduct (dtLastModif)
GO
CREATE INDEX EmplDept_dtLastModif ON amEmplDept (dtLastModif)
GO
```

［**dtLastModif**］フィールドがインデックス化されているかどうかを確かめるには、実行可能ファイル「adblog」を使用します。このファイルを使用すると、WHERE句で［**dtLastModif**］フィールドにフィルタを適用するSQLクエリの実行を検証できます。

Asset Managementデータベース用の調整 - Sybase ASEエンジン

クエリのFROM部分に多数のテーブルがあるため、クエリの実行に時間がかかるという内容がLOGファイルに表示される場合は、Sybase Query Optimizerアプリケーションを使用して、処理時間を短縮することが推奨されています。

「amdb.ini」ファイルの変更

SQLクエリの処理方法を改善するには、Asset Managementアプリケーションの「amdb.ini」ファイルに以下の行を追加することも可能です。

```
PostConnectSql= set forceplan on
```

以下の行は、**DB ASE COPPER**データベース用の「amdb.ini」ファイルの設定です。

```
[DB ASE COPPER]
PostConnectSql=set forceplan on
stmtcache=500
LongDesc=
Engine=Sybase
Location=COPPER
EngineLogin=itam
EnginePassword=78C6143D43925F46F924205FBB42F0FED21594428DDCAC641ED
76CDAA17050EA1A124254200200
ReadOnly=0
CacheDir=
CacheSize=5120000
Base=EDS
Owner=
TableSpace=
TableSpaceIndex=
AmApiDll=aamapi35.dll
UseNTSecurity=0
```

これらのパラメータは、Asset Managementデータベースサーバの「amdb.ini」ファイル内で設定されなければなりません。

Asset Managementアプリケーションのクライアント部分が、Connect-Itサーバにインストールされている場合は、同じSybaseデータベースに関連付けられた2つの別の接続を確立できます。

- 1番目の接続はオプション「PostConnectSql=set forceplan on」と「stmtcache=500」を使用します。

```
[DB ASE ConnectIt]
PostConnectSql=set forceplan on
stmtcache=500
LongDesc=
Engine=Sybase
Location=COPPER
EngineLogin=itam
EnginePassword=78C6143D43925F46F924205FBB42F0FED21594428DDCAC64
```

```
1ED76CDAA17050EA1A124254200200
ReadOnly=0
CacheDir=
CacheSize=5120000
Base=EDS
Owner=
TableSpace=
TableSpaceIndex=
AmApiDll=aamapi35.dll
UseNTSecurity=0
```

- 2番目のオプションはこれらのパラメータを使用しません。

```
[DB ASE COPPER ACGUI]
LongDesc=
Engine=Sybase
Location=COPPER
EngineLogin=itam
EnginePassword=78C6143D43925F46F924205FBB42F0FED21594428DDCAC64
1ED76CDAA17050EA1A124254200200
ReadOnly=0
CacheDir=
CacheSize=5120000
Base=EDS
Owner=
TableSpace=
TableSpaceIndex=
AmApiDll=aamapi35.dll
UseNTSecurity=0
```

従来のAssetCenterクライアント（Sybase ASE以外）で、オプション「stmtcache=500」を使用してはなりません。

従来のAssetCenterクライアントでの処理性能に問題がある場合は、以下のオプションのいずれかを使用できます。

- PostConnectSql=set forceplan on
- PostConnectSql=set table count 3
- PostConnectSql=set table count 2

データベース型コネクタを使用するシナリオの性能を向上させる

Sybaseネイティブ接続を使用するデータベース型コネクタの場合、「PostConnectSql=set forceplan on」オプションを詳細オプションで入力すると、SQLクエリ実行の性能が向上する可能性があります。

詳細オプションについては、マニュアル『**コネクタ**』の**「コネクタの設定」**の章、**「高度な設定」**の節の**「詳細オプション」**を参照してください。

11 処理レポート

処理レポートは、コネクタまたはマッピングボックスが、ドキュメントを取り込む度に生成することができるドキュメントです。

各処理レポートでは、ドキュメントが適切に処理されたかどうかを確認できます。正常に処理されたドキュメントとは、拒否された要素が全くないドキュメントのことを指します。

ソースコネクタが処理レポートを使用するためには、ソースコネクタは、**SuccessReport**ドキュメントタイプを使用可能なドキュメントタイプとして発行しなければなりません（InfraTools Desktop Discoveryコネクタでは、**DirectoryPoolerAction**と呼ばれています）。

ProcessReport処理レポートと、**SuccessReport**ドキュメントタイプ間のマッピングにより、ソースコネクタは、自分がドキュメントに転換したデータの抽出元のファイルに、特定のアクションを実行することができます。

［処理レポートの管理］ウィザードでは、処理レポートとソースコネクタの**SuccessReport**ドキュメントタイプ間のマッピングを、自動的に作成できます。

処理レポートの内容

処理レポートは、各コネクタとマッピングボックスが取り込むことができるドキュメントタイプとして存在します。処理レポートに含まれる情報の詳細は以下の表の通りです。

表 11.1. 処理レポートの内容

要素	情報	フィールドタイプ
ProcessReport	処理レポートのルートノード	
DocumentType	取り込み用ドキュメントタイプ	テキスト
DocumentTypeID	ドキュメントの固有識別子	テキスト
ErrorNumber	ドキュメントの取り込み中に発生したエラーの番号	整数（32ビット）
Success	ドキュメントの正常な処理（値：1）または正常でない処理（値：0） 正常でない処理は、ドキュメントの一部または全体の拒否を意味します。	ブール
WarningNumber	ドキュメントの取り込み中に発生する警告の数	整数（32ビット）
Logs	処理の失敗を説明するメッセージに対応するコレクション	
Date	メッセージの日付	日付と時刻
LogType	メッセージのタイプ 可能な値は以下の通りです。 • エラー • 警告 • 情報 • 追加情報	整数（32ビット）
Msg	メッセージの内容	テキスト
Path	メッセージに関連するドキュメントのパス	テキスト

処理レポートを使用できるコネクタ

コネクタが処理レポートを使用するためには、以下の条件が満たされていなければなりません。

- コネクタは、SuccessReportを使用可能なドキュメントタイプとして発行する。

  このドキュメントタイプは、処理レポート（ProcessReportドキュメントタイプ）内に含まれる要素をマップする役目を果たします。

  SuccessReportドキュメントタイプを使用できるコネクタは以下の通りです。
  - XMLコネクタ
  - テキストコネクタ
  - Peregrine Desktop Inventoryコネクタ
  - InfraTools Desktop Discoveryコネクタ
  - Eメールコネクタ（受信）
  - MQSeriesコネクタ
- コネクタの設定ウィザード内に［処理後のアクションを定義する］ページが含まれている。

  このページでは、コネクタがSuccessReportドキュメントを取り込んだ後の、ドキュメント処理後のアクションを指定できます。

  例：［処理後のアクションを定義する］ページでは、ターゲットコネクタが拒否した全ドキュメントを、「error」フォルダ内に移動させるようにコネクタを設定できます。

注意:

上記の2つの条件を満たさないコネクタも、処理レポートを使用できます。しかし、［処理レポートの管理］ウィザードを使用することはできません。

SuccessReportドキュメントタイプの内容

SuccessReportドキュメントタイプの内容はコネクタに応じて変化します。

ドキュメントタイプの詳細は以下の表の通りです。

コネクタ	要素	情報	フィールドタイプ
• XMLコネクタ • テキストコネクタ • PDIコネクタ	SuccessReport	処理レポートのルートノード	

	Success	ファイルの処理ステータス（成功または失敗）	ブール
	UrlFileInfo	処理されたドキュメントの情報を含む構造体	
	CreationDate	ファイル作成の日付	日付
	LastModificationDate	ファイルの最終変更日	日付
	Path	ファイルのパス	テキスト

コネクタ	要素	情報	フィールドタイプ
• InfraTools Desktop Discovery コネクタ	DirectoryPoolerAction	処理レポートのルートノード	
	FileName	FSFファイルの名前	テキスト
	Success	ファイルの処理ステータス（成功または失敗）	ブール

コネクタ	要素	情報	フィールドタイプ
• Eメールコネクタ（受信）	SuccessReport	処理レポートのルートノード	
	MailInfo		テキスト
	UniqueID	メッセージのID	整数

コネクタ	要素	情報	フィールドタイプ
• MQSeriesコネクタ	SuccessReport	処理レポートのルートノード	
	Success	ファイルの処理ステータス（成功または失敗）	ブール
	MessageInfo	処理されたメッセージに対応する構造体	
	MsgID	メッセージのID	整数
	PutDate	メッセージ処理の日時	日付

[処理後のアクションを定義する] ページを使用する

処理レポートは、ファイルからドキュメントを生成するコネクタが、ファイル処理の成功または失敗時にアクションを実行できるようにします。

- ファイルを元のフォルダに保存する
- ファイルを元のフォルダから削除する
- ファイルを別のフォルダに移動させる

  例：「error」フォルダ

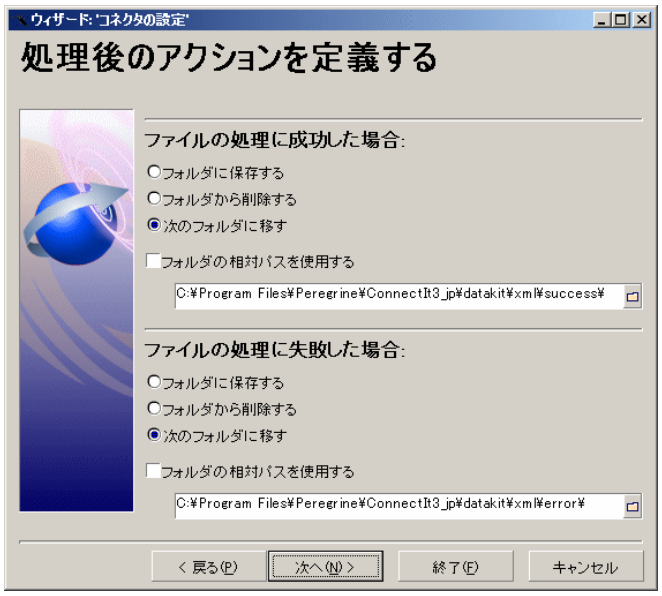
全コネクタ用に、以下の処理後のアクションがあります。

[処理後のアクションを定義する] ページを表示する

1. このページを使用できるコネクタを選択します。

   例：テキストコネクタ
2. [ツール／設定]（F2キー）を選択し、設定ウィザードを起動します。
3. [処理後のアクションを定義する] ページが表示されるまで [次へ] をクリックします。

   次のスクリーンショットはXMLコネクタの [処理後のアクションを定義する] ページです。



[処理レポートの管理] ウィザード

[処理レポートの管理] ウィザードでは、ソースコネクタの生成用ドキュメントタイプ用に以下の内容を作成します。

- このドキュメントタイプを取り込むターゲットコネクタと、ソースコネクタ間のフィードバックループ

  このフィードバックループは、ターゲットコネクタが生成する処理レポートの情報と、ソースコネクタのSuccessReportドキュメントタイプ間をマップします。
- 場合によっては、マッピングボックスとソースコネクタ間のフィードバックループ

[処理レポートの管理] ウィザードを使用する

1. シナリオビルダを起動します。
2. ソースコネクタがSuccessReportドキュメントタイプを発行するシナリオを開きます。

   SuccessReportを使用可能なドキュメントタイプとして発行するコネクタのリストについては、「処理レポートを使用できるコネクタ [p. 197]」の節を参照してください。
3. **[ツール／ウィザード／処理レポートの管理]** を選択します。
4. **[ソースドキュメントタイプを選択する]** ページで、どのドキュメントタイプ用にフィードバックループを作成するかを指定します。
5. **[ターゲットコネクタを選択する]** ページでは、前ページで選択されたソースドキュメントタイプの処理レポートを、どのターゲットコネクタ用に取得するかを指定します。
6. **[ソースコネクタにリンクしたマッピングボックスで処理レポートの作成を強制する]** を必要に応じて選択します。
7. **[終了]** をクリックします。

処理レポート - 使用例

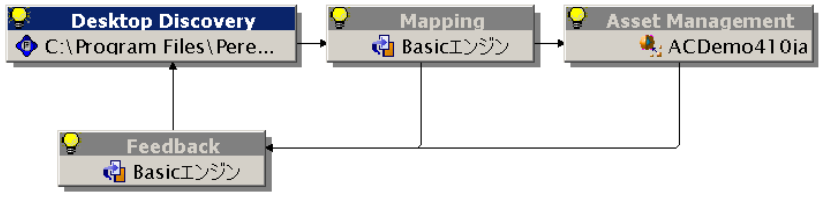
Connect-Itの付属シナリオでは、ソースコネクタに使用されるファイル上に実行するアクション（削除、移動）を決定するために、処理レポートが使用される場合もあります。

処理レポートをシナリオに統合するためには、まず処理レポートを使用している以下の付属シナリオを試行することをお勧めします。

次節では、idd \iddac36 \iddac.scnシナリオにおけるInfraTools Desktop Discoveryコネクタの処理レポートの使用方法を説明します。このシナリオでは、InfraTools Desktop Discoveryコネクタが処理するFSFファイルに基づいて、AssetCenterデータベース内でレコードを挿入／更新できます。

idd \iddac36 \iddac.scnシナリオでの処理レポートの使用

図 11.1. idd \iddac36 \iddac.scnシナリオ図



このシナリオでは、**feedback**というマッピングボックスは、Mappingマッピングボックスと Asset Managementコネクタにより生成された処理レポートを、InfraTools Desktop Discoveryコネクタに送ります。

このマッピングボックスの中には2つのマッピングがあります。

- **Mapping**マッピングボックスの処理レポートと、InfraTools Desktop Discoveryソースコネクタの**DirectoryPoolerAction**ドキュメントタイプ間のマッピング
- Asset Managementコネクタの処理レポートと、InfraTools Desktop Discoveryソースコネクタの**DirectoryPoolerAction**ドキュメントタイプ間のマッピング

表 11.2. Mappingマッピングボックスに生成された処理レポートと、InfraTools Desktop Discoveryコネクタに取り込まれるDirectoryPoolerActionドキュメントタイプ間のマッピングの詳細

DirectoryPoolerActionドキュメントタイプの要素	処理レポートの要素またはスクリプト

DirectoryPoolerAction（ルートノード）	if [Success] = 1 then
	PifIgnoreDocumentMapping
	end if

DirectoryPoolerActionドキュメントタイプの要素	処理レポートの要素またはスクリプト
コメント：ドキュメントが正常に取り込まれた場合、マッピングスクリプトは、マッピングボックスが処理レポートを生成しないようにします。 このシナリオでは、ドキュメントの処理の成功は、Asset Managementターゲットコネクタによるドキュメントの正常な処理に、従属しています。	
FileName	[$ParentDoc$.FileInfo.FileName]
InfraTools Desktop Discoveryコネクタにより生成される**Machine**ドキュメントタイプの**「FileInfo.FileName」**フィールドを取得できるようにします。このフィールドは、コネクタに処理されるFSFファイルの名前に一致します。	
Success	Success
コメント：ブールフィールド • 値「0」は、ドキュメントの一部または全部がマッピングボックスにより拒否されたことを意味します。 • 値「1」は、ドキュメントがマッピングボックスにより正常に処理されたことを意味します。	

表 11.3. Asset Managementコネクタに生成される処理レポートと、InfraTools Desktop Discoveryコネクタの取り込み用DirectoryPoolerActionドキュメントタイプ間のマッピングの詳細

DirectoryPoolerActionドキュメントタイプの要素	処理レポートの要素またはスクリプト
DirectoryPoolerAction（ルートノード）	
FileName	[$ParentDoc$.$ParentDoc$.FileInfo.FileName]
コメント：InfraTools Desktop Discoveryコネクタにより生成される**Machine**ドキュメントタイプの**「FileInfo.FileName」**フィールドを取得できるようにします。このフィールドは、コネクタに処理されるFSFファイルの名前に一致します。	
Success	Success
コメント：ブールフィールド • 値「0」は、ドキュメントの一部または全部がAsset Managementコネクタにより拒否されたことを意味します。 • 値「1」は、ドキュメントがAsset Managementコネクタにより正常に処理されたことを意味します。	

注意:

$ParentDoc$変数の使用

- 表記法［$ParentDoc$. フィールド］により、処理されるドキュメントの親ドキュメント内のフィールド値を、取得できるようになります。
- 表記法［$ParentDoc$.$ParentDoc$. フィールド］により、処理されるドキュメントの親ドキュメントの親ドキュメント内のフィールド値を、取得できるようになります。
- その他

InfraTools Desktop Discoveryコネクタに処理されたFSFファイルへの処理レポートの影響

処理レポートにより、InfraTools Desktop Discoveryコネクタは、処理レポートに関連するドキュメントに対応するFSFファイルに特定のアクションを実行できるようになります。処理される各ドキュメントの**FileInfo.FileName**フィールドで、ドキュメントに対応するFSFファイルが識別されます。

特定のアクションは、InfraTools Desktop Discoveryコネクタの設定時に選択できます。

例 :

FSFファイルの管理オプションは以下の通りです。

- ドキュメントの処理に成功した場合、ドキュメントに対応するFSFファイルは元のフォルダから削除されます。
- ドキュメントの処理に失敗した場合、ドキュメントに対応するFSFファイルは、例えば「エラー」と名前を付けたフォルダに移されます。

シナリオでドキュメントの処理を実行した後、「エラー」フォルダを開ければ、Asset Managementアプリケーションに適切に転送されなかったデータを含む全てのFSFファイルを、見つけることができます。

ドキュメントの一部または全部が拒否される時に起こった問題を確認するには、シナリオビルダのドキュメントログを参照してください。

Asset Managementコネクタにドキュメントが適切に処理されると、処理レポート（**Success**フィールドの値は「1」）は、InfraTools Desktop Discoveryコネクタが、ドキュメントに対応するFSFファイルを元のフォルダから削除するようにします。

入門プログラム

本節の入門プログラムでは、XMLソースコネクタとAsset Managementターゲットコネクタを使用するシナリオで、［処理レポートの管理］ウィザードを使用する方法を説明します。

この入門プログラムに必要なデータは、「[Connect-Itインストール先フォルダ] \datakit \tutorial \pro_rep」フォルダ内にあります。

前提条件

この入門プログラムを使用するには以下の条件を満たさなければなりません。

- 入門プログラムのフォルダにある「pro_rep.scn」シナリオを開く
- AssetCenter 4.1または4.2のテスト用データベースにアクセスできる

入門プログラム用シナリオの説明

入門プログラム用シナリオでは、

- 入門プログラムフォルダの「files」サブフォルダにある「.xml」ファイルを基に、XMLソースコネクタはドキュメントを生成します。
- マッピングボックスはこれらのドキュメントを取り込み、Asset Managementコネクタが取り込めるようにドキュメントを生成します。
- Asset Managementコネクタは、AssetCenterのテスト用データベースの［amAbsence］テーブル内にレコードを作成します。

XMLコネクタとAsset Managementコネクタ間のマッピング

XMLコネクタのamAbsenceドキュメントタイプと、Asset ManagementコネクタのamAbsenceドキュメントタイプ間のシナリオのマッピングは、以下の表の通りです。

ターゲットドキュメントタイプの要素	ソースドキュメントタイプの要素	マッピングスクリプト

amAbsence	amAbsence	if [Nature] = "" then
		PifRejectDocumentMapping("Document rejected: missing nature")
		end if
		このスクリプトによって、XMLコネクタが生成するドキュメントのNatureフィールドが空の場合、ドキュメントを拒否できるようになります。

AccountingType	AccountingType	AccountingType
Nature	Nature	Nature (識別キー)
PhoneContact	PhoneContact	PhoneContact (識別キー)

入門プログラム用ファイル

入門プログラム用フォルダの「files」サブフォルダには、以下の「.xml」ファイルが含まれています。

- success.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<amAbsence>
 <AccountingType>Convention</AccountingType>
 <bArchived>1</bArchived>
 <dtBack>2002-06-14 18:00:00</dtBack>
 <dtLastModif>2002-07-12 05:30:32</dtLastModif>
 <dtOut>2002-06-14 09:00:00</dtOut>
 <dtValidation>2002-01-15 18:00:00</dtValidation>
 <fDays>1</fDays>
 <Field1>Test</Field1>
 <Nature>Meeting</Nature>
 <PhoneContact>06.12.11.81</PhoneContact>
 <Reason>San Diego Convention</Reason>
 <seValidated>1</seValidated>
</amAbsence>
```

シナリオの全コンポーネントはこのファイルの処理に成功するはずです。またシナリオのテスト後、このファイルは「success」フォルダに移されます。

- src_fail.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<amAbsent>
 <AccountingType>Convention</AccountingType>
 <bArchived>1</bArchived>
 <dtBack>2002-06-14 18:00:00</dtBack>
 <dtLastModif>2002-07-12 05:30:32</dtLastModif>
 <dtOut>2002-06-14 09:00:00</dtOut>
 <dtValidation>2002-01-15 18:00:00</dtValidation>
 <fDays>1</fDays>
 <Field1>Field1</Field1>
 <Nature>Meeting</Nature>
 <PhoneContact>06.12.11.81</PhoneContact>
 <Reason>San Diego Convention</Reason>
 <seValidated>1</seValidated>
</amAbsent>
```

このファイルはXMLコネクタに拒否されます。これは**amAbsent**構造が、コネクタに使用される「pro_rep.dtd」DTD内で定義されている構造に一致しないためです。

シナリオのテスト後に、このファイルは「error」フォルダに移されます。

- map_fail.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<amAbsence>
 <AccountingType>Convention</AccountingType>
 <bArchived>1</bArchived>
 <dtBack>2002-06-14 18:00:00</dtBack>
 <dtLastModif>2002-07-12 05:30:32</dtLastModif>
 <dtOut>2002-06-14 09:00:00</dtOut>
 <dtValidation>2002-01-15 18:00:00</dtValidation>
 <fDays>1</fDays>
 <Field1>Test</Field1>
 <Nature></Nature>
 <PhoneContact>06.12.11.81</PhoneContact>
 <Reason>San Diego Convention</Reason>
 <seValidated>1</seValidated>
</amAbsence>
```

このファイルはマッピングボックスに拒否されます。**<Nature>**要素が空であるためです。

シナリオのテスト後に、このファイルは「error」フォルダに移されます。

- dest_fail.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<amAbsence>
 <AccountingType>Convention</AccountingType>
 <bArchived>1</bArchived>
 <dtBack>2002-06-14 18:00:00</dtBack>
 <dtLastModif>2002-07-12 05:30:32</dtLastModif>
 <dtOut>2002-06-14 09:00:00</dtOut>
 <dtValidation>2002-01-15 18:00:00</dtValidation>
 <fDays>1</fDays>
 <Field1>Test</Field1>
 <Nature>Meeting</Nature>
 <PhoneContact></PhoneContact>
 <Reason>San Diego Convention</Reason>
 <seValidated>1</seValidated>
</amAbsence>
```

このファイルはAsset Managementコネクタに拒否されます。識別キー（コネクタの取り込み用ルール）として選択された**<PhoneContact>**要素が空であるためです。

シナリオのテスト後に、このファイルは「error」フォルダに移されます。

手順1 - 用例シナリオのXMLコネクタの設定

1. シナリオのXMLコネクタを選択します。
2. **［F2］**キーを押します。
3. **［処理モードを選択する］**ページで**［読み取り］**オプションを選択します。
4. **［接続のプロトコルを選択する］**ページで**［ローカルファイル／ネットワークファイル］**オプションを選択します。
5. **［ファイルまたはフォルダを選択する］**ページで、
   1. **［フォルダを読み取る］**オプションを選択します。
   2. をクリックし、入門プログラム用フォルダの「files」サブフォルダの場所を指定します。
6. **［処理後のアクションを定義する］**ページの2つの枠で、**［次のフォルダに移す］**オプションを選択します。

   ［ファイルの処理に成功した場合］には、「success」サブフォルダのパスを入力します。

   ［ファイルの処理に失敗した場合］には、「error」サブフォルダのパスを入力します。
7. **［DTD／XSDを選択する］**ページでは、入門プログラム用フォルダの「pro_rep.dtd」DTDのパスを指定します。
8. **［終了］**をクリックします。

手順2 - 用例シナリオのAsset Managementコネクタの設定

1 シナリオのAsset Managementコネクタを選択します。
2 **［F2］**キーを押します。
3 **［接続用パラメータを定義する］**ページで、AssetCenterテスト用データベースの接続パラメータを入力します。
4 **［終了］**をクリックします。

手順3 - ［処理レポートの管理］ウィザードの使用

1 **［ツール／ウィザード／処理レポートの管理］**を選択します。
2 **［ソースドキュメントタイプを選択する］**ページで、**amAbsenceSrc**ソースドキュメントタイプを選択します。
3 **［次へ］**をクリックします。
4 **［ターゲットコネクタを選択する］**ページでAsset Managementコネクタを選択します。
5 **［詳細オプション］**ページで、**［ソースコネクタにリンクしたマッピングボックスで処理レポートの作成を強制する］**を選択します。
6 **［終了］**をクリックします。

ウィザードは、XMLコネクタとAsset Managementコネクタ間にマッピングボックスを作成します。このマッピングボックス内ではマッピングが作成されます。マッピングは以下の表の通りです。

XMLコネクタの取り込み用ドキュメントタイプSuccessReportの要素	Asset Managementコネクタが生成する処理レポートの要素
Success	Success
コメント：ブールフィールド • 値「0」は、ドキュメントがAsset Managementコネクタにより拒否されたことを意味します。 • 値「1」は、ドキュメントがAsset Managementコネクタにより正常に処理されたことを意味します。	
UrlFileInfo	[$ParentDoc$.$ParentDoc$.UrlFileInfo.Path]
XMLコネクタの生成用ドキュメントタイプ**amAbsence**の**UrlFileInfo.Path**フィールドを取得できるようにします。このフィールドは、XMLコネクタに読み取られる「.xml」ファイルの名前に当たります。	

手順4 - ［処理レポートの管理］ウィザードで作成されたフィードバックループのテスト

1. XMLコネクタを選択します。
2. ▶をクリックします。

   シナリオ図では、⛔アイコンがシナリオのコンポーネント上に表示されるはずです。このアイコンは、「.xml」ファイル処理の失敗を意味します。シナリオビルダの［ドキュメントログ］タブを選択し、生成されたエラーメッセージを確認してください。
3. 「success」サブフォルダを開き、「success.xml」ファイルがあるかどうか確認します。
4. 「error」サブフォルダを開き、「dest_fail.xml」、「map_fail.xml」と「rc_fail.xml」ファイルがあるかどうか確認します。

12 Connect-ItのJava開発キット（JDK）

Connect-ItのJava開発キットを使用すると、ユーザは独自のコネクタを開発できます。

この開発キットは、XMLテクノロジとJava Connector Architecture （JCA）1.0規格を使用しています。

この規格の詳細については、Webサイト http://java.sun.com/j2ee/connector/ を参照してください。

Java開発キットの内容

Connect-ItのJava開発キットのパッケージ内容は以下の表の通りです。

表 12.1. Connect-ItのJava開発キットの内容

パッケージ	機能
com.peregrine.common.assert	アサーション機構（デバッグモード）
com.peregrine.common.collection	コレクションのクラス

パッケージ	機能
com.peregrine.common.ids	STRファイル処理用のツールを提供します。アプリケーションのストリングの場所を見つけるのに便利です。
com.peregrine.common.io	ファイル管理用の相対クラス
com.peregrine.common.log	ログファイル（LOG）のメッセージを作成し配信する機構を適用します。
com.peregrine.common.text	文字列のフォーマット用ツール
com.peregrine.common.thread	スレッドの管理
com.peregrine.common.util	多種のツール
com.peregrine.common.xml	XML操作
com.peregrine.conit.core	Connect-Itフレームワークの中核
com.peregrine.conit.document	Connect-Itが使用するUnified Document Content規格
com.peregrine.conit.document.description	Connect-It XMLメタデータ
com.peregrine.conit.document.schema	XMLSchemaサポートのAPI
com.peregrine.conit.ra	JCAコネクタ
com.peregrine.resourceimpl.cci	CCIクラスの実装
com.peregrine.resourceimpl.event	イベントクラスの実装（EVENT）
com.peregrine.resourceimpl.spi	SPIクラスの実装
com.peregrine.resourceimpl.tools	導入ツール
com.peregrine.resourceimpl.util	JCA実装用のヘルパー

Java環境に関する情報

Java環境に関する情報を取得するには、

1. **［ヘルプ／バージョン情報］**メニューを選択します。
2. **［詳細］**をクリックします。

   Java環境に関する情報がウィンドウに表示されます（［Java］ノード）。

JCA規格の実装

本節は以下の3点について説明します。

- CCI（Common Client Interface）クラスの実装
- SPI（Service Provider Interface）クラスの実装
- EVENTクラスの作成によるJCA規格の拡張

CCIクラスの実装

インタフェース	javax.resource.cci.Connection
実装されるクラス	com.peregrine.resourceimpl.cci.APrgnConnectionHandle

インタフェース	javax.resource.cci.ConnectionFactory
実装されるクラス	com.peregrine.resourceimpl.cci.PrgnConnectionFactory

インタフェース	javax.resource.cci.ConnectionMetaData
実装されるクラス	com.peregrine.resourceimpl.spi.APrgnManagedConnectionMetaData

インタフェース	javax.resource.cci.Interaction
実装されるクラス	com.peregrine.resourceimpl.cci.APrgnXMLInteraction

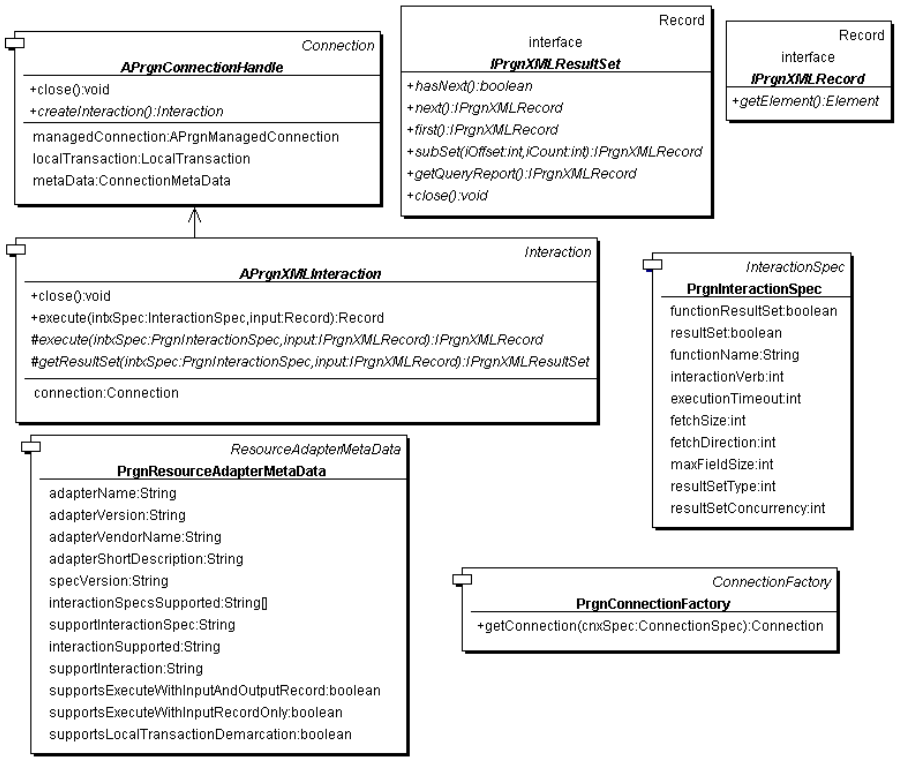
インタフェース	javax.resource.cci.InteractionSpec
実装されるクラス	com.peregrine.resourceimpl.cci.PrgnInteractionSpec

インタフェース	javax.resource.cci.ResourceAdapterMetaData
実装されるクラス	com.peregrine.resourceimpl.cci.PrgnResourceAdapterMetaData

インタフェース	javax.resource.cci.Record
実装されるクラス	com.peregrine.resourceimpl.cci.IPrgnXMLRecord com.peregrine.resourceimpl.cci.IPrgnXMLResultSet

インタフェース	javax.resource.cci.LocalTransaction
実装されるクラス	com.peregrine.resourceimpl.spi.APrgnLocalTransaction

図 12.1. Java開発キット - CCIクラス



本節ではJCA規格の主要なクラスを説明します。

javax.resource.cci.Connection

このクラスは、物理的な接続を確立するために必要なクライアントのアプリケーションレベルのハンドルを表しています。

実装

APrgnConnectionHandleクラスは抽象的なクラスです。このクラスにより、ManagedConnectionオブジェクト経由で、物理的な接続へアクセスできるようになります。ManagedConnectionインスタンスが作成するブリッジにより、異種のクライアント間での物理的なリソースの共有が可能になります。これら2レベルの一貫性を保つために、接続の確認機構が付属しています。

javax.resource.cci.ConnectionFactory

これはクライアントコンポーネントのエントリポイントで、getConnection()の2メソッドのうち1つを経由して接続を確立します。

実装

PrgnConnectionFactoryは、即時に使用可能な具体クラスです。非管理環境では、このクラスはPrgnNonManagedConnectionManagerオブジェクトに代表されるConnectionManagerインタフェースのインスタンスを、デフォルトで作成します。

javax.resource.cci.ConnectionMetaData

このクラスは、Connectionタイプの接続経由でシステムに関する情報を提供します。これはManagedConnectionMetaData SPIインタフェースクライアントと対をなします。

実装

選択された基本的実装は「インタフェースプログラミング」契約に基づきます。抽象クラスAPrgnManagedConnectionMetaDataは、cciとspiインタフェースで記述された役割をまとめます。

javax.resource.cci.LocalTransaction

このクラスはアプリケーションレベルのトランザクションを表します。このインタフェースは、SPIインタフェースLocalTransactionに関連しています。

実装

cciとcpiの役割は、Service Provider Interfaceパッケージのベースクラス APrgnLocalTransaction内にまとめられています。

javax.resource.cci.ResourceAdapterMetaData

このクラスはコネクタに対応する情報を説明します。

実装

PrgnResourceAdapterMetaDataクラスは具体クラスで、インタフェースに記述されるプロパティのアクセッサ（getXXXとsetXXXメソッド）を提供します。

javax.resource.cci.Record

このクラスは、システムとの相互作用時の入力データと出力データを表します。

実装

実装フレームワークにサポートされているRecordタイプの全オブジェクトは、XML規格に基づいています。以下の2つのタイプがあります。

- IPrgnXMLRecord

単一XMLドキュメントに相当します。

- **IPrgnXMLResultSet**
  XMLドキュメント全体のイテレータに相当します。

SPIクラスの実装

インタフェース	javax.resource.spi.ConnectionEventListener
実装されるクラス	com.peregrine.resourceimpl.spi.APrgnConnectionEventListener

インタフェース	javax.resource.spi.ConnectionManager
実装されるクラス	com.peregrine.resourceimpl.spi.PrgnNonManagedConnectionManager

インタフェース	javax.resource.spi.ConnectionRequestInfo
実装されるクラス	com.peregrine.resourceimpl.spi.PrgnConnectionRequestInfo

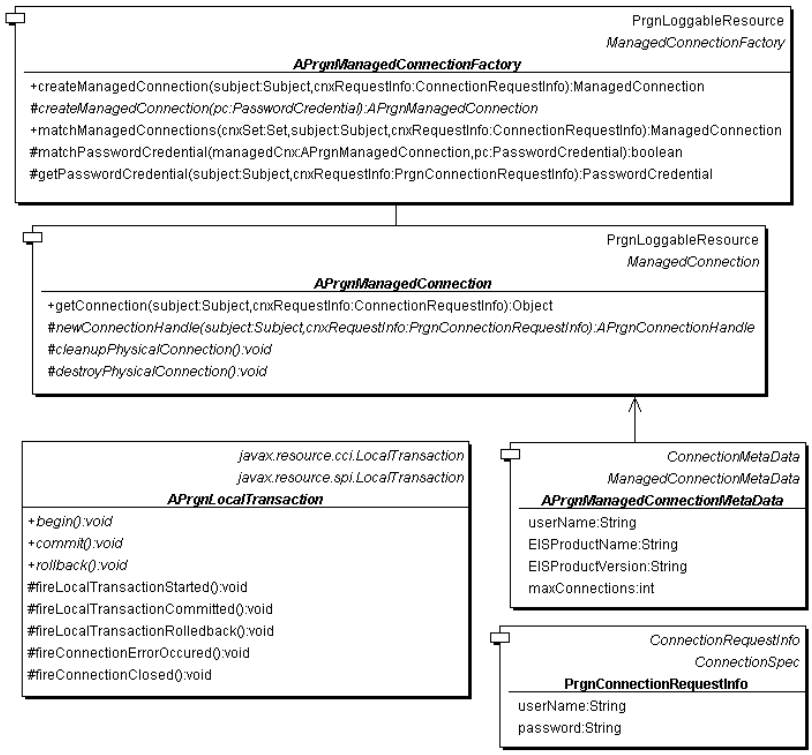
インタフェース	javax.resource.spi.LocalTransaction
実装されるクラス	com.peregrine.resourceimpl.spi.APrgnLocalTransaction

インタフェース	javax.resource.spi.ManagedConnection
実装されるクラス	com.peregrine.resourceimpl.spi.APrgnManagedConnection

インタフェース	javax.resource.spi.ManagedConnectionFactory
実装されるクラス	com.peregrine.resourceimpl.spi.APrgnManagedConnectionFactory

インタフェース	javax.resource.spi.ManagedConnectionMetaData
実装されるクラス	com.peregrine.resourceimpl.spi.APrgnManagedConnectionMetaData

図 12.2. Java開発キット - SPIクラス



javax.resource.spi.ConnectionEventListener

このクラスは、クライアント接続に関するManagedConnectionインスタンスの通知を受信できるようにします。

実装

抽象クラスAPrgnConnectionEventListenerは、インタフェースで記述されるメソッドを実装します。これらのメソッドは空のため、複数の適切な機能をオーバーロードできます（design pattern 'adapter'）。

javax.resource.spi.ConnectionManager

プールやトランザクションを管理するために、コネクタとアプリケーションサーバをリンクできるようにします。

実装

デフォルトの具体的な実装はコネクタ（非管理環境では必須）に提供されています。プール機能は追加されます。

javax.resource.spi.ConnectionRequestInfo

接続要求のフローの一部である特定のクライアントの、プロパティの集合です。

実装

PrgnConnectionRequestInfoは、**ユーザ名**と**パスワード**のプロパティをサポートします。特定のコネクタを実装すると他のプロパティも追加できます。

javax.resource.cci.ConnectionSpecクラスの実装も可能にします。

javax.resource.spi.LocalTransaction

ローカルトランザクションの管理をサポートできるようにします。

実装

APrgnLocalTransactionは、APrgnManagedConnectionインスタンスにリンクしており、トランザクション通知用のメソッドも提供します。

javax.resource.spi.ManagedConnection

EISインスタンスの物理的な接続を表します。

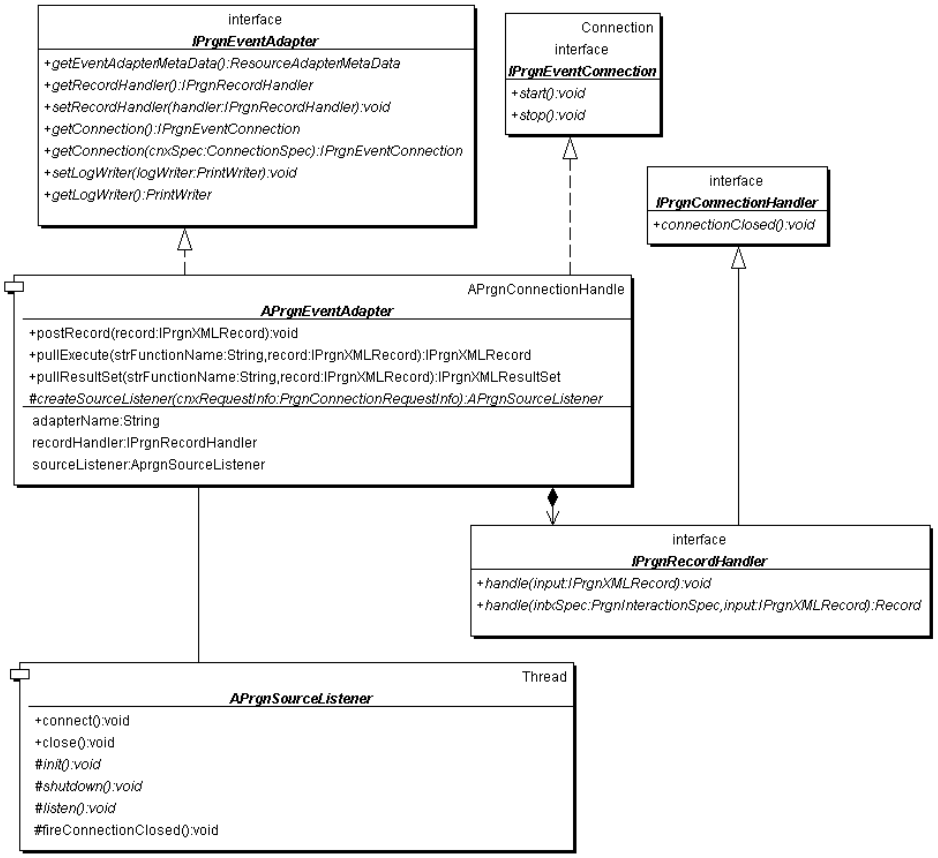
実装

APrgnManagedConnectionは、接続ハンドルの集合を維持しつつ接続の共有をサポートします。

JCAの拡張：イベントクラス

JCA規格は非同期の作動をサポートしません。このイベントモードを使用するためには、EISがコールバックの相互作用と通知を管理できなければなりません。この種の相互作用は、以下の表にあるクラスを使って生成されます。

図 12.3. Java開発キット - イベントクラス



com.peregrine.resourceimpl.event.IPrgnEventAdapter

```
//Event adapter metadata
javax.resource.cci.ResourceAdapterMetaData getEventAdapterMetaData()

//EIS event listener registration
IPrgnRecordHandler getRecordHandler()
void setRecordHandler(IPrgnRecordHandler handler)

//connection methods
IPrgnEventConnection getConnection() throws javax.resource.ResourceException
IPrgnEventConnection getConnection(javax.resource.cci.ConnectionSpec cnxSpec) thr
```

```
ows javax.resource.ResourceException

//logging methods
void setLogWriter(PrintWriter logWriter) throws ResourceException
PrintWriter getLogWriter() throws ResourceException
```

com.peregrine.resourceimpl.event.IPrgnEventConnection

```
//event treatment methods
void start() throws javax.resource.ResourceException
void stop() throws javax.resource.ResourceException
```

com.peregrine.resourceimpl.event.IPrgnRecordHandler

```
//Connection notification
void connectionClosed();
```

UDC（Unified Document Content）規格

Connect-Itでは、コネクタの生成用または取り込み用ドキュメントタイプに合致したXMLドキュメントの形で、データが伝達されます。

ドキュメントタイプの構造は、Connect-It専用のUDC（Unified Document Content）規格に準拠します。

UDC規格はドキュメントタイプ内で3つの要素（またはノード）を定義します。

- **<STRUCTURE>**

  この要素は、コネクタの使用可能なドキュメントタイプ内の構造体に相当します。
- **<COLLECTION>**

  この要素は、コネクタの使用可能なドキュメントタイプ内のコレクションに相当します。
- **<ATTRIBUTE>**

  この要素は、コネクタの使用可能なドキュメントタイプ内の属性に相当します。

要素	サポートされる従属要素	コンテンツの混合をサポート	データコンテンツをサポート	コンテンツの基数
<COLLECTION>	• <COLLECTION> • <STRUCTURE> • <ATTRIBUTE>	• いいえ	• いいえ	Unbounded
<STRUCTURE>	• <COLLECTION> • <STRUCTURE> • <ATTRIBUTE>	• はい	• いいえ	Bounded
<ATTRIBUTE>	• いいえ	• いいえ	• はい	N/A

<ATTRIBUTE>要素にサポートされているデータ型は以下の通りです。

- Boolean
- Char
- Short
- Integer
- Long
- Float
- Double
- String
- Date
- Time
- Timestamp
- Blob
- Memo

Javaコネクタの使用可能なドキュメントタイプは、この規格に従っています。使用可能な各ドキュメントタイプのルートノードは、<STRUCTURE>要素に相当し、<STRUCTURE>要素の名前は、コネクタの接続先アプリケーションの1データ集合に当たります。例：このデータ集合は、データベース型コネクタではテーブルの名前になります。

JDBC Javaコネクタの使用可能なドキュメントタイプの例

```
<STRUCTURE name="myTable">
 <ATTRIBUTE name="field1" type="String"/>
 <ATTRIBUTE name="field2" type="Long"/>
 <COLLECTION name="link">
  <STRUCTURE name="myLinkedTable">
   <ATTRIBUTE name="field3" type="String"/>
   <ATTRIBUTE name="field4" type="Memo"/>
  </STRUCTURE>
```

```
 </COLLECTION>
</STRUCTURE>
```

1ドキュメントの全ノードの名前は固有でなければなりません。例外は以下の場合です。

- ノードの複数フィールド（<ATTRIBUTE>要素）が同じデータを含む場合
- ノードが統括する<COLLECTION>と<STRUCTURE>要素が同じである場合

UDC規格の実装

Connect-ItのJava開発キットには、XMLとUDCドキュメントの編集用に多数のクラスが付属しています。

クラスは、パッケージcom.peregrine.conit.documentにまとめられています。

- PrgnDocument
  XML文書の一般的な定義を統括します。
- APrgnElement
  UDCラッパの親クラス
- PrgnAttribute
  <ATTRIBUTE>ノードのラッパ
- APrgnElementContainer
  <STRUCTURE>と<COLLECTION>ノードの親クラス
- PrgnStructure
  <STRUCTURE>ノードのラッパ
- PrgnCollection
  <COLLECTION>ノードのラッパ

ドキュメントタイプはPrgnStructureラッパと共に実装されなければなりません。

これらのクラスの詳細については、「[Connect-Itインストール先フォルダ]\doc\javadoc\conit\index.html」ファイルをクリックし、HTML文書Javadocを参照してください。

Javaコネクタの作成

本節では、Connect-ItでのJavaコネクタの作成方法を説明します。

前提条件

Javaコネクタの作成方法は、外部アプリケーション（EIS）の作動モードに応じて変化します。

外部アプリケーション（EIS）には、大きく分けて以下の2つの作動モードがあります。

- 要求／応答モード（例：リレーショナルデータベース）
- イベント／リッスン（例：メッセージシステム）

本節では、要求／応答モードを使用するJavaコネクタの作成について説明します。イベントコネクタの説明については、本章の「イベントコネクタを作成する [p. 240]」の節を参照してください。

接続の契約を定義する

接続の契約を定義する場合、以下の内容を定義します。

- 外部アプリケーション（EIS）の接続タイプ
- この接続を確立するためクライアントコンピュータに必要なパラメータ

  これらのパラメータはXML導入ファイル（例：ra.xml）の<config-property>ノードに対応します。

接続タイプ

物理的な接続は、 **com.peregrine.resource.impl.spi.APrgnManagedConnection**派生クラスに含まれています。

以下のメソッドを書く必要があります。

- cleanupPhysicalConnection(): void

  このメソッドは以下の内容を実装しなければなりません。
  - プールされた接続が送信された時点で実行するアクション
  - 次回使用時用の接続の準備
- destroyPhysicalConnection(): void

  このメソッドは、外部アプリケーション（EIS）への接続の切断を実装しなければなりません。

また、物理的な接続を経由して外部アプリケーション（EIS）のメタデータへアクセスするためのコードを、以下の方法で作成する必要もあります。

- **com.peregrine.resourceimpl.spi.APrgnManagedConnectionMetaData**を継承するクラスを書く
- このクラスのインスタンスを返すために**getMetaData() : ManagedConnectionMetaData**メソッドをオーバーロードする

接続作成のパラメータ

APrgnManagedConnectionFactoryクラスの役割は、外部アプリケーション（EIS）へ物理的な接続を作成することです。

導入ファイル内で定義される接続の全プロパティでは、物理的な接続を作成するために、データのget（取得）やset（初期化）のパブリックメソッドを作成する必要もあります（例：データベース名、サーバ名など）。

コネクタのメタデータをエクスポーズするためには、getMetaData() : ResourceAdapterMetaDataメソッドをオーバーロードする必要もあります。これによって、サポートされる相互作用のタイプを知ることができます。このメソッドは、com.peregrine.resourceimpl.cci.PrgnResourceAdapterMetaDataのインスタンスを戻さなければなりません。

クライアントの接続を取得するために必要なパラメータ

これらのJCAクラスを使用するコードは、PrgnConnectionRequestInfoオブジェクトを用いて接続を要求します。これは**「ユーザ名」**と**「パスワード」**の接続パラメータを使用するコンポーネントです。コードのこの場所に、ユーザ接続の取得用に必要な追加パラメータを置くことができます。

トランザクションの契約を定義する

トランザクションの契約を定義するということは、外部アプリケーション（EIS）がトランザクションをサポートするということを意味します。

注意:

Connect-Itはローカルトランザクションのみをサポートします。

トランザクションの契約を定義するには、com.peregrine.resourceimpl.spi.APrgnManagedConnectionクラスの、getLocalTransaction() : LocalTransactionメソッドを使用する必要があります。

このメソッドがサポートされていない場合は、例外javax.resource.NotSupportedExceptionを起動します。サポートされている場合は、APrgnLocalTransactionの派生クラスを書いて、以下のメソッドを使用します。

- begin() : void
  新規のトランザクションを開始します。
- commit() : void
  トランザクションをコミットします。
- rollback() : void
  トランザクションをロールバックします。

ステータス通知のメソッドを使用することも可能です。

- fireLocalTransactionStarted() : void

  管理されリンクされた接続に、ローカルトランザクションが始動したことを通知します。
- fireLocalTransactionCommited() : void

  管理されリンクされた接続に、ローカルトランザクションがコミットされたことを通知します。
- fireLocalTransactionRolledBack() : void

  管理されリンクされた接続に、ローカルトランザクションがロールバックされたことを通知します。

セキュリティの契約を定義する

セキュリティの契約を定義すると、外部アプリケーション（EIS）へのアクセスをセキュリティで保護できるようになります。

ベースクラス**APrgnManagedConnectionFactory**は、セキュリティで保護されたコンテキスト内で物理的な接続を作成します。

デフォルトの基本実装は、認証機構**BasicPassword**です。多くの場合この実装で十分です。

認証用に使用されるセキュリティ情報は、**javax.resource.spi.security.PasswordCredential**オブジェクトです（ユーザ名と、パスワード）。

getPasswordCredential(Subject, ConnectionRequestInfo) メソッドは、Factory用に**javax.security.auth.Subject**オブジェクトからセキュリティ情報を抽出します。サブジェクトがない場合（Connect-Itはこの場合に当たります）、メソッドは作成された**PasswordCredential**を経由して、**com.peregrine.resourceimpl.spi.PrgnConnectionRequestInfo**オブジェクト内に含まれた情報を直接使用します。

以下のメソッドを作成する必要があります。

createManagedConnection(PasswordCredential) : APrgnManagedConnection

別の認証機構が必要な場合、1番高いレベルのメソッドをオーバーロードする必要があります。

createManagedConnection(Subject, ConnectionRequestInfo) : ManagedConnection

派生クラス**APrgnManagedConnection**内に、次のメソッドをオーバーロードします。

newConnectionHandle(Subject, ConnectionRequestInfo) : APrgnConnectionHandle

新規接続への参照を作成するには、

- クライアントが、外部アプリケーション（EIS）の物理的インスタンス上での再認証をサポートしない場合は、セキュリティ情報を無視しなければなりません。サポートする場合は、**APrgnManagedConnectionFactory**と同じセキュリティ機構を使用します。
- 再認証がサポートされている場合、コネクタは**APrgnManagedConnection**インスタンスのセキュリティコンテキストと、配置された**Subject**インスタンスのコンテキストを変更します。

データを定義する

Connect-Itで、コネクタはXML文書（ドキュメント）の形でデータを交換します。これらのドキュメントの内容は、Unified Document Content（UDC）規格に基づいています。各ドキュメントは1ドキュメントタイプに基づいており、ドキュメントタイプは更にUDC規格に準拠しています。ドキュメントは、1構造体（<STRUCTURE>ノード）で表現され、構造体はフィールド（<ATTRIBUTE>ノード）、構造体（<STRUCTURE>ノード）とコレクション（<COLLECTION>ノード）を含みます。

UDC規格の詳細については、本章の「UDC（Unified Document Content）規格 [p. 220]」の節を参照してください。

データへの相互作用を定義する

データへの相互作用を定義すると、JCA規格のクライアント部分が実行する演算を定義できるようになります。最も重要なのは、外部アプリケーション（EIS）のデータにアクセスするメソッドに関連します。

Connect-Itのコネクタの相互作用は、XMLの相互作用です。ベースコンポーネントは、**com.peregrine.resourceimpl.APrgnXMLInteraction**クラスと共に、相互作用をサポートします。XML相互作用のメソッドの結果（1つまたは複数）は、以下の2つのタイプになります。

- XMLRecord
- XMLResultset

コネクタのベースコンポーネントは、設定、接続、自動記述などのために相互作用を定義します。これらの演算は全コネクタに共通しており、変更されたりオーバーロードされたりしてはなりません。

このため、コネクタ独自の相互作用には同じ名前を使用することができません。使用してはならない名前は以下の通りです。

- **connect**
- **disconnect**
- **start**

- stop
- setConfig
- getConfig
- getOperations
- describe
- getRecords
- viewRecords
- getRequests
- getRequest
- setEventFilter
- startTransaction
- endTransaction
- commitTransaction
- prepareTransaction
- rollbackTransaction
- forgetTransaction
- recoverTransaction
- reconc
- getDocumentChildrenName
- getDocumentsName

コネクタ独自の演算を定義するためには、com.peregrine.conit.ra.APrgnDocumentInteractionクラスが提供されています。これは、APrgnXMLInteractionの派生クラスです。このクラスの機能は、コネクタのXML演算として相互作用のメソッドをカプセル化することです。このクラスは更に、これらのメソッドの保存もサポートします。

3種類の演算があります。

- status

  「status」演算は、取り込みモードにおけるコネクタの機能に相当します。この演算は「データ」（data）型のパラメータを取得し、このデータの処理ステータスを戻します。例：「失敗」または「成功」

  例：データベース型のコネクタでは、「insert」は「status」タイプの演算です。

- resultSet

  「resultSet」演算は、生成モードにおけるコネクタの機能に相当します。この演算は「データスキーマ」（ドキュメントタイプ）型のパラメータを取得し、データ集合を戻します。

例：データベース型コネクタで、「query」はresultSetタイプの演算です。この演算はテーブル名と、クエリの対象となるフィールドのリストをパラメータとして取得し、対応するレコードを戻します。

- **document**

  この演算については、Connect-Itの開発チームが後日執筆するテクノートを参照してください。

具体メソッドのタイプは、以下の3つのパブリック署名に従う必要があります。

- **method(PrgnDocument in) : PrgnMessage**

  status型の演算
- **method(PrgnDocument in) : IPrgnXMLResultSet**

  resultSet型の演算
- **method(PrgnDocument in) : PrgnDocument**

  document型の演算

コネクタの自動記述

クライアント部分が以下の情報を取得できるようにするため、コネクタは自動記述しなければなりません。

- コネクタの生成用または取り込み用ドキュメントタイプ
- 生成用または取り込み用のドキュメントタイプにコネクタが実行できる演算
- コネクタが適用できるルール（ディレクティブ）
- コネクタが使用するスケジュールのポインタ

コネクタの自動記述は、**com.peregrine.conit.document.IPrgnDescribable**インタフェースを介して実行されます。このインタフェースは、**com.peregrine.conit.ra.APrgnDocumentInteraction**クラスに実装されており、以下のメソッドを定義します。

- **getSubFormats(PrgnPath path, String strMode) : PrgnDocument**
- **getOperations(): Vector**
- **getDirectives() : Vector**
- **getPointers() : Vector**

これらのメソッドを作成すると、コネクタは自動記述できるようになります。**APrgnDocumentInteraction**に実装されるデフォルトの動作は、ドキュメントタイプも演算もないコネクタです。

Connect-Itの全コネクタに共通の**describe**演算は、この自動記述を検索できるようにします。

使用可能なドキュメントタイプ - Javaコネクタの自動記述

Javaコネクタの使用可能なドキュメントタイプを定義するには、**com.peregrine.conit.ra.APrgnDocumentInteractionクラスの、 getSubFormat(PrgnPath path, String strMode) : PrgnDocumentメソッドをオーバーロードする必要があります。**

strModeパラメータは、ドキュメントタイプのどの記述が必要であるかを指定します。このパラメータは以下の値をとります。

- resultSet

  または、生成用ドキュメントタイプに対応する com.peregrine.conit.core.IPrgnOperation.RESULTSET
- status

  または、取り込み用ドキュメントタイプに対応する com.peregrine.conit.core.IPrgnOperation.STATUS
- document

  または、com.peregrine.conit.core.IPrgnOperation.DOCUMENT

注意:

このパラメータについては、Connect-Itの開発チームが後日執筆するテクノートを参照してください。

---

getSubFormatメソッドは、原則として外部アプリケーションのデータの単層記述（非階層型な記述）を戻します。完全な記述は、継続呼び出しによって取得されます。

パラメータ「path」が空の場合、メソッドは取り込み用または生成用で使用可能なドキュメントタイプのリストを戻さなければなりません。データベース型のコネクタでは、空の「path」パラメータはこの種のリストを戻すことがあります。

```
<schema>
 <STRUCTURE name="docType1"/>
 <STRUCTURE name="docType2"/>
 <STRUCTURE name="docType3"/>
 <STRUCTURE name="docType4"/>
 (...)
</schema>
```

注意:

返されるXMLドキュメントのルートノードに特に意味はありません。また、有効なXML要素であれば全てルートノードになり得ます。

「path」パラメータが空でない場合、メソッドは、このパラメータ値が指定するドキュメントタイプの1子ノードの内容を戻さなければなりません。パスが無効であると例外が発生する可能性があります。

例1

pathパラメータ値として「docType2」を使用する

```
<schema>
 <ATTRIBUTE name="a1" type="Long"/>
 <ATTRIBUTE name="a2" type="Byte"/>
 <STRUCTURE name="s1"/>
 <COLLECTION name="c2"/>
 <STRUCTURE name="s2/>
 (...)
</schema>
```

例2

pathパラメータ値として「docType2.s1」を使用する。この場合、ドキュメントタイプ「docType2」の子ノード「s1」が、取得されるドキュメントです。

```
<schema>
 <ATTRIBUTE name="a5" type="String"/>
 <COLLECTION name="c1"/>
 (...)
</schema>
```

注意:

- 生成用または取り込み用にコネクタがドキュメントタイプをエクスポーズしない場合、NULLが戻されることもあります。
- PrgnPathクラスは、パスの要素上で処理を繰り返すパブリックサブクラスPrgnPathTokenizerを含みます。

演算 - Javaコネクタの自動記述

演算は以下の要素で定義されます。

- 名前
- 演算タイプ
- 演算を使用できるドキュメントタイプのリスト

コネクタは、取り込みまたは生成モードで複数の演算をエクスポーズできます。

Javaコネクタの演算をエクスポーズするには、com.peregrine.conit.ra.APrgnDocumentInteractionクラスの、getOperations() : Vectorメソッドをオーバーロードする必要があります。

このメソッドは、com.peregrine.conit.document.description.PrgnOperationDescのベクタを戻さなければなりません。戻されるベクタが空であると、コネクタは演算を使用できません。**注意**：イベントコネクタのgetRecords演算の場合は、この限りではありません。

PrgnOperationDescクラスコンストラクタでは以下のパラメータが必要です。

- 1演算名
- 1演算タイプ

  このパラメータは文字列であり、status演算（com.peregrine.conit.core.IPrgnOperation.STATUSクラス）、またはresultSet演算（com.peregrine.conit.core.IPrgnOperation.RESULTSETクラス）になり得ます。

PrgnOperationDesc.addParam(String strParam) :voidメソッドでは、演算にパラメータを追加できます。strParamパラメータは、この演算の適用先となるドキュメントタイプの名前に当たります。

1つの演算は複数のドキュメントタイプに適用されることもあります。デフォルトで、1演算は全ドキュメントタイプに適用されます。

注意:

statusとresultSet型の演算は、statusとresultSet型のドキュメントタイプにそれぞれ適用されます。

シナリオビルダで、コネクタは演算のリストをコネクタのルール内に表示します。

演算のルール（ディレクティブ）- Javaコネクタの自動記述

コネクタのルール（ディレクティブ）は、コネクタがサポートする演算を分類します。

例：データベース型のコネクタでは、ソースデータベースにフィルタを適用するWHERE句とORDERBY句は、取り込み用ルールに当たります。

シナリオビルダでコネクタのルールを入力する方法については、マニュアル『**コネクタ**』の「**コネクタのルール（ディレクティブ）**」の章を参照してください。

ルールとルールグループ

ルールは機能ごとにまとめられています。例えば、整合性チェック用ルールのグループなどです。

ルールの定義

ルール（ディレクティブ）は以下の要素で定義されます。

- 名前
- タイプ
- デフォルト値（オプション）
- ルールの適用先となるノードタイプのリスト
- シナリオビルダ内でルールのタイトルを構成する記述（オプション）

ルールは、以下のXMLドキュメントのノードに適用されることができます。

- **<STRUCTURE>**

  この要素は、コネクタの使用可能なドキュメントタイプ内の構造体に相当します。

  ドキュメントタイプのルートノード（root）は、別の方法で処理される**<STRUCTURE>**です。
- **<COLLECTION>**

  この要素は、コネクタの使用可能なドキュメントタイプ内のコレクションに相当します。
- **<ATTRIBUTE>**

  この要素は、コネクタの使用可能なドキュメントタイプ内の属性に相当します。

ルールグループの定義

ルールグループは以下の要素により定義されます。

- 名前
- ルールのリスト
- ルールグループを使用する演算のリスト
- シナリオビルダ内でルールグループのタイトルを構成する記述（オプション）

データベース型のコネクタが使用するルールグループ「FILTER」と「RECONCILIATION」の例は、以下の表の通りです。

グループ名	演算	ルール（2）	タイプ	デフォルト値	関連する導入ファイルのノード
FILTER	query	WHERE	Memo	なし（空のメモ）	ルートノードに適用される（データベースのテーブル）
		ORDERBY	文字列型	なし（空の文字列）	

グループ名	演算	ルール（1）	タイプ	デフォルト値	関連する導入ファイルのノード
RECONCILIATION	insert	RECONCKEY	ブール型	false	ATTRIBUTEノードに適用される（データベースのフィールド）

ルールの値の処理

演算が呼び出されると、ルールの値は、パラメータとして渡されたXMLドキュメントの<layer>要素内に渡されます。

例

```
<operation>
 <STRUCTURE name="format">
  <ATTRIBUTE name="a1" type="String"/>
  <ATTRIBUTE name="a2" type="Long"/>
  <STRUCTURE name="s1">
   <ATTRIBUTE name="sa1" type="Boolean"/>
  </STRUCTURE>
 </STRUCTURE>
 <layer>
  <STRUCTURE name="format">
   <dir1>value</dir1>
   <dir2>value</dir2>
   <ATTRIBUTE name="a1" type="String">
    <dir3>value</dir3>
   </ATTRIBUTE>
   <STRUCTURE name="s1">
    <dir2>value</dir2>
```

```
      <ATTRIBUTE name="sa1" type="Boolean">
        <dir3>value</dir3>
      </ATTRIBUTE>
    </STRUCTURE>
  </STRUCTURE>
 </layer>
</operation>
```

ルールの値は、コネクタの使用可能なドキュメントタイプの特定のノードに関連付けられています。**<layer>**要素は、コネクタの使用可能なドキュメントタイプの構造を複製します。この際、これらのフィールドの値（**<ATTRIBUTE>**要素）や、ルールが全く適用されていないノード、またはルールの値がルールのデフォルト値と同一であるノードは、複製されません。

com.peregrine.conit.ra.APrgnDocumentInteractionクラス内の、**getDirectiveValue(PrgnDocument doc, PrgnPath path, String strDirectiveName, int iDirectiveType) : Object**メソッドは、ルールの値を取得します。

docパラメータは、演算のパラメータとして渡されたXMLドキュメントを含まなければなりません。

pathパラメータは、どのノードでルールの値を取得するかを指定します。このパスに、ドキュメントタイプのルートノードが含まれてはなりません。

pathDirectiveTypeh パラメータは、ルールの基本タイプを指定します。拡張タイプのルールでは、**PrgnTypes.STRING**タイプを渡します。

戻されるオブジェクトクラスは、ルールの基本タイプにより変化します。ルールの値が見つからない場合は、**NULL**値が戻されます。

以下のコードの例では、「**Format.s1.sa1**」ノードの「**dir3**」値を検索します。**dir3**のタイプは「**PrgnTypes.INT**」であると仮定します。

例

```
public PrgnDocument operation(PrgnDocument doc) throws ResourceException
{
 (...)
 PrgnPath path = new PrgnPath();
 path.addRight("s1").addRight("sa1");
 int iVal = ((Integer)getDirectiveValue(doc, path, "dir3", PrgnTypes.INT)).intValue();
 (...)
}
```

ルールグループの表示

ルールグループをエクスポーズするには、**com.peregrine.conit.ra.APrgnDocumentInteraction**クラスの、**getDirectiveGroups() : Vector**メソッドをオーバーロードする必要があります。

このメソッドは、com.peregrine.conit.document.description.PrgnDirectiveGroupDescのVectorを戻します。戻されたVectorが空である場合、コネクタで使用可能なルールはありません。

PrgnDirectiveGroupDescクラスコンストラクタには、パラメータが1つだけあります（ルールグループの名前）。

ルールグループを使用する演算を定義するには、以下のメソッドを使用します。

- addOperation(PrgnOperationDesc desc) : void
- addOperation(String strOperationName, String strOperationType) : void

グループにルールを追加するには、addDirectiveDesc(PrgnDirectiveDesc desc) :voidメソッドを使用します。

com.peregrine.conit.document.description.PrgnDirectiveDescクラスコンストラクタには2つのパラメータがあります。

- ルールの名前
- ルールのタイプ

ルールのタイプは、このルール用に予期されている値のタイプを決定します。例：ブール型ルールで、使用可能な値は「true」または「false」です。このタイプは、ルール編集のグラフィカルコントロールも指定します。例：「ファイル名」タイプのルールにアイコン付きのフィールドを使用する。

可能なルール（ディレクティブ）のタイプは、com.peregrine.conit.document.PrgnTypesクラス内で定義されています。

これらのタイプに、com.peregrine.conit.document.description.PrgnDirectiveDescクラス内で定義される拡張タイプを追加する必要があります。

- ファイル名
  TYPE_FILE_NAME
- フォルダ名
  TYPE_DIR_NAME
- 変更不可能なリストデータ
  TYPE_ENUM
- 変更可能なリストデータ
  TYPE_FREE_ENUM

注意:

全ての拡張タイプの基本タイプは、PrgnTypes.STRINGです。

---

setDefaultVal(Object) :voidメソッドは、ルールのデフォルト値を定義できるようにします。パラメータとして渡されるオブジェクトクラスは、ルールのタイプに対応しなければなりません。またルールの拡張タイプの場合は、オブジェクトクラスが基本タイプに対応しなければなりません。このデフォルト値は必須ではありません。

ルールの適用先のノードのタイプを定義するには、以下のメソッドを使用します。

- setApplyOnRoot(boolean bApply) : void
- setApplyOnStructures(boolean bApply) : void
- setApplyOnCollections(boolean bApply) : void
- setApplyOnAttributes(boolean bApply) : void

デフォルトでは、ルールはどのノードにも適用されません。

addPossibleValue(Object) : voidとsetPossibleValues(Collection) : voidメソッドは、以下の内容を定義します。

- ルールのタイプがPrgnDirectiveDesc.TYPE_ENUM（変更不可能なリストデータ）の場合は、ルールの使用可能な値
- ルールのタイプがPrgnDirectiveDesc.TYPE_FREE_ENUM（変更可能なリストデータ）の場合は、ルールに既に設定された値

スケジュールのポインタ - Javaコネクタの自動記述

シナリオがスケジュールモードで起動している場合、スケジュールのポインタとはソースコネクタが使用する一定の値を指します。ポインタは、最終セッションと同一のデータが処理されるのを回避するために使用されます。

スケジュールのポインタの概念は、resultSetタイプの演算でのみ有用です。

データベース型コネクタでは、スケジュールのポインタは、前回のセッション以降に変更または作成されたレコードのみを処理するために使用されます。この場合スケジュールのポインタは、一般的にレコードの最終変更の日付を記すフィールドになります。

キュー内に格納されるデータを処理するコネクタ（MQSeriesコネクタ）の場合、スケジュールのポインタは、最終セッション後に作成されたデータ集合のみを処理するために使用されます。データ集合は複数のメッセージ、文書、XMLファイルなどに相当します。この場合、スケジュールのポインタはデータ集合の番号に当たります。**例**：第1回目のセッションで、MQSeriesコネクタが番号0001から0999までのメッセージを処理したとします。第2回目のセッションでは、0999より大きな番号のメッセージのみが処理されます。

スケジュールのポインタの定義

ポインタは、以下の要素により定義されます。

- ポインタのデータ型
- デフォルト値

  この値は、スケジュールモードでシナリオの最初のセッションが起動する時の値です。
- ドキュメントタイプのリスト

このリストは、ドキュメントタイプをまとめており、ドキュメントタイプの生成はこのスケジュールのポインタに左右されます。

ポインタ値の処理

resultSetタイプの演算がスケジュールモードで呼び出されると、ポインタ値はpointerルールの値として渡されます。

com.peregrine.conit.ra.APrgnDocumentInteractionクラスで、getPointerValue(PrgnDocument doc) : Objectメソッドはポインタ値を取得します。

docパラメータは、演算のパラメータとして渡されたXMLドキュメントを含まなければなりません。

戻されるオブジェクトクラスは、ポインタのタイプにより変化します。以下の2つの場合はNull値が戻されます。

- スケジュールのポインタが定義されていない。
- シナリオがスケジュールモードで起動していない。

新規ポインタ値は、ルールのレポートを介してConnect-Itへ伝達されます。ルールは、getQueryReport() : IPrgnXMLRecordメソッドにより作成されます。レポートはXMLドキュメントの形を取ります。

新規ポインタ値は、以下の2つの属性を含む<pointer>ノード（<pointers>ノード内に含まれる）に渡されます。

- 'type'
  ポインタのタイプ
- 'name'
  現在のドキュメントタイプ名

クエリのレポートの例

```
<report>
 (...)
 <pointers>
  <pointer name="format" type="String">new pointer value</pointer>
  (...)
 </pointers>
 (...)
</report>
```

getQueryReportの実装例

```
public IPrgnXMLRecord getQueryReport() throws ResourceException
{
 PrgnDocument report = new PrgnDocument("report");
 (...)
 PrgnDocument pointers = report.createChildDocument(IPrgnTags.POINTERS);
```

```
 PrgnDocument pointer = pointers.createChildDocument(IPrgnTags.POINTER);
 pointer.setAttribute(IPrgnTags.ATTR_NAME, "tableName");
 pointer.setAttribute(IPrgnTags.ATTR_TYPE, PrgnTypes.getTypeName(PrgnTypes.LONG));
 PrgnXMLUtil.addTextNode(pointer.getElement(), m_pointerValue.toString());
 (...)
 return report;
}
```

注意:

新規ポインタ値がレポートに渡される場合に、シナリオがスケジュールモードで起動していないと、この値は無視されます。

スケジュールのポインタ値は、Connect-Itログ内に保存されます。

スケジュールのポインタのステータス（値）を確認するには、以下の手順に従います。

1. シナリオビルダを起動します。
2. 生成モードで、シナリオまたはコネクタを開きます。
3. **［シナリオ／スケジュール］**を選択します。

スケジュールのポインタのエクスポーズ

スケジュールのポインタをエクスポーズするには、**com.peregrine.conit.ra.APrgnDocumentInteraction**クラスの、**getPointers() : Vector**メソッドをオーバーロードする必要があります。

このメソッドは**com.peregrine.conit.document.description.PrgnPointerDesc**クラスの**Vector**を戻さなければなりません。戻された**Vector**が空であると、コネクタはスケジュールのポインタを使用しません。

PrgnPointerDescクラスコンストラクタには、パラメータが1つだけあります。これは「ポインタのタイプ」です。

ポインタを使用するドキュメントタイプを定義するには、以下のメソッドを使用します。

- addSchema(String strDocTypeName) : void
- addSchemas(Collection cDocTypeNames) : void

デフォルトで、スケジュールのポインタは全ドキュメントタイプに適用されます。

Java仮想マシン（JVM）を設定する

Javaコネクタ用にConnect-Itが使用するJVMは、シナリオ図で直接設定可能です。

JMVの設定方法

1 シナリオビルダを起動します。
2 **［Java／JVMを設定する］**を選択します。
3 **［クラスパス］**フィールドにJavaクラスパスを入力します（jarまたはzipファイル）。
   複数のパスを指定する場合は、セミコロン（;）を使用します。
4 **［オプション］**フィールドにJVMオプションを入力します。
   例：**-DmyDefine=value**
5 **［デバッグモード］**オプションをオンまたはオフにします。
   このオプションにより、デバッグのメッセージ（**PrgnLog.debug**）と例外のスタックトレース（**stacktrace**）を表示できるようになります。

Javaコネクタの導入

Javaコネクタを導入するには、コネクタを動的に導入するXML導入ファイル（多くの場合ra.xmlファイル）が必要です。このファイルは、JCA規格またはこの規格のイベント拡張に基づいています。導入ファイルの例については、本章の「イベントコネクタを作成する [p. 240]」の節の「Javaコネクタの導入 - イベントコネクタ [p. 242]」を参照してください。

このファイルを作成した後に、コネクタをConnect-It内に導入します。

1 Connect-Itを起動します。
2 **［Java／コネクタを導入する］**を選択します。
3 をクリックします。
4 コネクタに名前を付けます。
   ［コネクタ名］フィールドは必須で、これはツールボックス内のコネクタの名前になります。

5 コネクタの導入ファイルを指定します。
6 コネクタのタイプを選択します。
以下の選択肢があります。
- **イベント**
  JCA規格のイベント拡張
- **リソース**
  JCA規格の実装

7 ツールボックス内のコネクタ用のアイコンを選択します。
このアイコンは16ピクセルx16ピクセルの寸法でなければなりません。
8 **［識別キー］**フィールドで、取り込み用ドキュメントタイプの編集ウィンドウに識別キーを表示するコネクタのルールを指定します。
例：付属のJDBCコネクタで、このルールは**RECONCKEY**ルールです。
Javaコネクタのルールについては、本章の「コネクタの自動記述 [p. 228]」節の「演算のルール（ディレクティブ）- Javaコネクタの自動記述 [p. 231]」を参照してください。
9 オプションを選択します。
- **取り込みモード**
  コネクタがドキュメント取り込みの演算（書き込み、送信など）をサポートする場合は、このオプションを選択します。
- **生成モード**
  コネクタがドキュメント生成の演算（読み取り、受信など）をサポートする場合は、このオプションを選択します。
- **スケジュールモードでポインタを使用する**
  コネクタがスケジュールのポインタを使用する場合は、このオプションを選択します。
  Javaコネクタでのスケジュールのポインタの実装については、本章の「コネクタの自動記述 [p. 228]」節の「スケジュールのポインタ - Javaコネクタの自動記述 [p. 236]」を参照してください。
- **コネクタは要求を発信する**
  コネクタが別のコネクタに向けて要求を送信する場合は、このオプションを選択します。例：XMLリスニングコネクタ、SCAutoリスニングコネクタ。このオプションは、イベントコネクタのみで使用可能です。

イベントコネクタを作成する

本節では、イベントコネクタの作成方法を段階を追って説明します。

トランザクションの契約を定義する - イベントコネクタ

com.peregrine.resourceimpl.event.IPrgnEventAdapterインタフェースは、コネクタアーキテクチャの主要なエントリポイントです。これは、接続契約とセキュリティに関してIPrgnEventAdapterクラスの役割を担当し、またgetConnection()メソッドを使用します。このため具体的な実装は、導入ファイルで定義された設定パラメータに対応する接続を作成する際に必要なパラメータ用に、GetとSetメソッドを提供する必要があります。

デフォルトで提供されている実装はcom.peregrine.resourceimpl.event.APrgnEventAdapterです。この抽象クラスは、接続インタフェースを実装し、接続の操作を可能にします。start()とstop()関数は、イベント処理のコントロールをイベント接続上に追加します。

クライアントが接続を発生させると（getConnection()メソッド）、外部アプリケーションの特定のイベントリスナが、createSourceListener()メソッドにより動的に作成されます。作成されたオブジェクトcom.peregrine.resourceimpl.event.APrgnSourceListenerは、java.lang.Threadのように動作し、外部アプリケーションの以下のタスクを実行します。

- 接続の初期化

  connect()とinit()メソッド
- 接続の切断

  disconnect()とshutdown()メソッド
- 外部アプリケーションから来る入力イベントのリッスンと、XMLレコードの生成

  listen()メソッド
- 接続イベントの通知

  addConnectionHandler()と fireConnectionClosed()メソッド

ManagedConnectionFactoryのgetMetaData()メソッドのように、getEventAdapterMetaData()メソッドはコネクタのメタデータを返します。このメソッドでは、アクティブな接続が外部アプリケーションと確立される必要はありません。

データを定義する - イベントコネクタ

イベントタイプのコネクタはUnified Document Content規格をサポートします。

イベントコネクタ用のデータの定義は、他のJavaコネクタと同じです。データの定義については、本章の「Javaコネクタの作成 [p. 222]」の節の「データを定義する [p. 226]」を参照してください。

データへの相互作用を定義する - イベントコネクタ

IprgnEventAdapterのもう1つの役割は、生成されるイベントと要求のターゲットオブジェクトを保存することです。このオブジェクトはcom.peregrine.resourceimpl.event.IPrgnRecordHandlerインタフェースを実装し、応答せずにXMLファイルを取り込むことができます（イベントの受信）。また、入力レコードから出力レコードを送信する再呼び出しの相互作用として、動作できます（要求）。

保存されたハンドラへのXMLイベントのルーティングは、postRecord()メソッドを経由して実行されます。要求はpullExecute()とpullResultSet()メソッドを経由して実行されます。

相互作用のカスタムメソッドの書き込みも、イベントモードでは許可されています。しかしこれらのメソッドは、要求／応答モードでの機能（クライアントの要求によるドキュメントの生産または取り込み）の一部です。

データでの相互作用の詳細については、本章の「Javaコネクタの作成 [p. 222]」の節の「データへの相互作用を定義する [p. 226]」を参照してください。

自動記述 - イベントコネクタ

イベントコネクタの自動記述は、他のJavaコネクタの自動記述と同じです。

コネクタの自動記述の詳細については、本章の「Javaコネクタの作成 [p. 222]」の節の「コネクタの自動記述 [p. 228]」を参照してください。

イベントコネクタは、イベントを生成したり要求を発信したりするために、演算をエクスポーズする必要はありません。エクスポーズする必要があるのは、カスタマイズされた演算のみです。

イベントコネクタが生成するイベントのフォーマットは、コネクタの生成用ドキュメント（resultSetモード）に一致します。

イベントコネクタは、カスタマイズされたstatusタイプの演算をサポートしない限り、ドキュメントタイプを取り込むことはありません。

Javaコネクタの導入 - イベントコネクタ

Javaイベントコネクタを導入するには、このコネクタを動的に導入するXML記述子（ra.xmlファイル）が必要です。

この記述子はJCA規格に類似しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<connector>
 <display-name> My Event Adapter< /display-name>
 <description> Event Adapter< /description>
```

```
 <vendor-name> Peregrine Systems< /vendor-name>
 <eis-type> My EIS Type< /eis-type>
 <version> 1.0< /version>
 <license>
   <description> < /description>
   <license-required> true< /license-required>
 </license>
 <resourceadapter>
   <eventadapter-interface> com.peregrine.resourceimpl.event.IPrgnEventAdapter< /ev
entadapter-interface>
   <eventadapter-class> MyEventAdapterClass< /eventadapter-class>
   <config-property>
     <description> < /description>
     <config-property-name> MyProperty< /config-property-name>
     <config-property-type> MyPropertyType< /config-property-type>
     <config-property-value> MyPropertyValue< /config-property-value>
   </config-property>
     ...
 </resourceadapter>
</connector>
```

JCAコネクタの作成 - 例

本節では、JDBCタイプの用例コネクタの作成方法を段階を追って説明します。

Java用例コネクタ用のソースは以下のフォルダにあります。

- [Connect-Itインストール先フォルダ] \datakit \javasample \com \peregrine \sample \ra \jdbc

接続の契約を定義する - 用例コネクタ

コネクタの接続パラメータは以下の通りです。

- **Driver**
  リレーショナルDBMSにアクセスするために使用するJDBCドライバ
- **URL**
  次のフォームを使います： <Subprotocol> : <DatabaseName>
- **UserName**
  接続作成用に使用される最初のセキュリティ情報
- **Password**
  接続作成用に使用される2番目のセキュリティ情報

このパラメータは、com.peregrine.resourceimpl.spi.APrgnManagedConnectionFactoryクラスを拡張できます。

```
public class JDBCManagedConnectionFactory extends APrgnManagedConnectionFact
ory
{
// class must be public to be deployed by an external tool.
}
```

コードのこの部分は、接続パラメータ用にデータのget（取得）とset（初期化）パブリックメソッドを取得できるようにします。

```
/**
  * Sets the JDBCManagedConnectionFactory URL property.
  */
 public void setURL(String strURL)
 {
?
 }

 /**
  * Gets the JDBCManagedConnectionFactory URL property.
  */
 public String getURL()
 {
   ?
 }
```

物理的な接続を考慮に入れるためには、com.peregrine.resourceimpl.spi.APrgnManagedConnectionクラスを派生させる必要があります。

トランザクションの契約を定義する - 用例コネクタ

用例コネクタはトランザクション境界をサポートしません。

ManagedConnectionFactoryクラスのgetResourceMetaDataメソッドは、falseに初期化されているResourceMetaDataオブジェクトを、supportsLocalTransactionDemarcationプロパティと共に戻さなければなりません。

ManagedConnectionの次のメソッドは、NotSupportedExceptionを生成しなければなりません。

- getLocalTransaction() : LocalTransaction
- getXAResource() : XAResource()
- getAutoCommit() : boolean

- setAutoCommit(boolean b) : void

セキュリティの契約を定義する - 用例コネクタ

セキュリティ機構は、**ユーザ名／パスワード**の認証に基づいています。

ManagedConnectionFactoryセキュリティに関連しているメソッドは以下の通りです。

```
createManagedConnection(,PasswordCredential pc) : APrgnManagedConnection
matchPasswordCredential(APrgnManagedConnection managedCnx, PasswordCredential pc) : boolean
```

再認証はサポートされていません。

ManagedConnectionセキュリティに関連しているメソッドは以下の通りです。

```
newConnectionHandle(Subject subject, PrgnConnectionRequestInfo cnxRequestInfo) : APrgnConnectionHandle
```

データを定義する - 用例コネクタ

用例コネクタは、リレーショナルデータベース型の外部アプリケーションに、SELECTを実行します。

データへの操作はXML技術に基づいているため、1演算は以下の形式の1ドキュメントを取り込みます。

```
<query>
 <STRUCTURE name="Employee">
  <ATTRIBUTE name="LastName">
  <ATTRIBUTE name="FirstName">
 </STRUCTURE>
</query>
```

このドキュメントの結果は、SQL宣言（SELECT LastName,FirstName FROM Employee）となります。この宣言は、データベースのレコードに操作を施す**XMLResultSet**を戻します。

ManagedConnectionFactoryの実装時に、以下のメソッドを追加する必要があります。

```
public ResourceAdapterMetaData getMetaData() throws javax.resource.ResourceException
 {
// must return a PrgnResourceAdapterMetaData instance
 PrgnResourceAdapterMetaData raMetaData = new PrgnResourceAdapterMetaData();
```

```
 ?
return raMetaData;
}
```

データへの相互作用を定義する - 用例コネクタ

コードはAPrgnDocumentInteractionを派生させ、次のパブリックメソッドを書きます。

```
query(PrgnDocument document) : IPrgnXMLResultSet
```

設定ウィザードを改善する

「ra.xml」ファイルは、Connect-Itシナリオビルダでコネクタを展開します。またこのファイルは、非常に単純なデフォルトの設定ウィザードを作成できるようにします。

コネクタの設定ウィザードの使用については、マニュアル『**コネクタ**』の「**コネクタの設定**」の章を参照してください。

本節では、デフォルトのウィザードを改善するDSDファイルを作成し、ウィザードに以下の複雑な機能を統合する方法を説明します。

- 複数選択肢のリスト
- デフォルト値の提示
- ラジオボタン（オプションボタン）
- オペレーティングシステムのフォルダ内の検索ボタン
- パスワードの暗号化入力ゾーン
- その他

警告:

無効なDSDファイルを使用すると、コネクタの実行に失敗することがあります。

DSDファイル導入前に作成された既存のシナリオは正常に起動します。しかし、DSDファイルと共に保存されたシナリオは、そのDSDファイルが削除されると起動不可能になります。

表記法：本節のコードの例で、赤はDSD内に常に存在する要素を表します。これらの要素は変化しません。

青は、必須の属性を表します。

DSDファイルの作成

DSDファイルは、コネクタの「ra.xml」ファイルを含むフォルダ内に挿入されなければなりません。

例：コネクタの名前が「MyCon」である場合、コネクタを導入すると、対応するフォルダがConnect-Itインストール先フォルダ内に作成されます（[Connect-Itインストール先フォル]//config/java_MyCon」）。

DSDファイルは、導入ファイルと同名でなければなりません。

例：導入ファイルの名前が「MyCon.xml」である場合、DSDの名前は「MyCon.dsd」になります。

DSDファイルの内容

ファイルには以下の5つのノードが含まれなければなりません。

- **<DSD>ノード**

  このノードの属性は「rootnode」で、以下の4つのノードをまとめています。
- **<module>ノード**

  このノードは、全DSDファイルで同一です。
- 3つの**<node>必須ノード**
  - 第1のルートノードは、設定のプロパティの宣言を含みます。

    このノードの「name」属性の値は、**<DSD>**ノードの「rootnode」属性の値と同じでなければなりません。

    「class」属性と「edit」属性は必須であり、値はそれぞれ「Folder」と「CWBAutoDet」でなければなりません。
  - defaultノードとadvancedノードの属性は「name」であり、属性の値はそれぞれ「default」と「advanced」です。

    これらの2ノードは、単純モードと高度なモードの両方でウィザードのページを定義します。これらのノードの「class」属性の値は、ルートノードの「name」属性の値でなければなりません。

例

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DSD name="Wiz" rootnode="ConWiz" version="1.0" warning="true">
 <!-- You must specify the root node -->
<module name="core.dsd" local="true"/>
<!-- This module node is mandatory -->
 <node name="ConWiz" class="Folder" edit="CWBAutoDet">
 (...)
 </node>
 <node name="default" class="ConWiz">
```

```
(...)
</node>
<node name="advanced" class="ConWiz">
(...)
</node>
</DSD>
```

advancedノードがdefaultノードと相違しない場合、以下の要領で単純にすることもできます。

```
<node name="advanced" class="default"/>
```

警告:

「name」属性はタグの第1属性でなければなりません。

コネクタ設定のプロパティの定義

コネクタの設定のプロパティごとに、ルートノードには子ノードが含まれなければなりません。この子ノードは、導入ファイル（ra.xml）で使用される名前と同じ名前になります。

警告:

ルートノードで定義される全プロパティは、導入ファイル内に含まれなければなりません。

必須属性

設定の各プロパティには2つの必須属性「name」と「class」があります
サポートされるクラスは以下の通りです。

- Enum（変更不可能なリストデータ）
- FreeEnum（変更可能なリストデータ）
- Boolean（ブール）
- Long（倍長整数）
- Double（倍精度）
- Date（日／時）
- Duration（期間）
- String（文字列）
- Password（暗号化）
- FileName（ファイル名）

- DirName（フォルダ名）
- Memo（複数行のテキスト）

オプション属性

設定の各プロパティには、3つのオプション属性があります。

- 'mandatory'

  ブール型
- 'inshortdesc'

  このブール値がtrueである場合、パラメータの値はコネクタの要約記述内で使用されます。
- 'default'

  この属性は、設定プロパティのデフォルト値を定義します。

属性「inshortdesc」または「mandatory」が指定されていない場合、デフォルト値は「false」になります。

設定のプロパティ

各設定プロパティには2つの子ノードがあります。

- <shortdesc>

  このプロパティは、設定ウィザードがシナリオビルダ内で起動される時にウィザードに表示されるフィールドの説明を定義します。このプロパティは必須です。
- <longdesc>

  このプロパティは、コネクタのヘルプ用テキストを定義します。

例

```
<node name="ConWiz" class="Folder" edit="CWBAutoDet">
 <node name="field1" class="String" default="test" mandatory="true" inshortdesc="tru
e">
<shortdesc>Field 1</shortdesc>
<longdesc>Field 1's help</longdesc>
 </node>
 <node name="field2" class="Boolean" default="true" mandatory="false" inshortdesc="
false">
<shortdesc>Field 2</shortdesc>
<longdesc>Field 2's help</longdesc>
 </node>
 <node name="field3" class="Long" default="0">
<shortdesc>Field 3</shortdesc>
<longdesc>Field 3's help</longdesc>
```

```
</node>
<node name="field4" class="Double" default="1.52">
<!-- Double nodes have a precision of two digits after the decimal point.-->
<shortdesc>Field 4</shortdesc>
<longdesc>Field 4's help</longdesc>
</node>
<node name="field5" class="Date" default="yyyy-mm-dd">
<shortdesc>Field 5</shortdesc>
<longdesc>Field 5's help</longdesc>
</node>
<node name="field6" class="Duration" default="1y 2mon 3d 4h 5m 6s">
<shortdesc>Field 6</shortdesc>
<longdesc>Field 6's help</longdesc>
</node>
<node name="field7" class="Password">
<shortdesc>Field 7</shortdesc>
<longdesc>Field 7's help</longdesc>
</node>
<node name="field8" class="FileName" default="c:/tmp/test.txt">
<shortdesc>Field 8</shortdesc>
<longdesc>Field 8's help</longdesc>
</node>
<node name="field9" class="DirName" default="c:/tmp">
<shortdesc>Field 9</shortdesc>
<longdesc>Field 9's help</longdesc>
</node>
<node name="field10" class="Enum" default="op1">
<shortdesc>Field 10</shortdesc>
<longdesc>Field 10's help</longdesc>
<enum name="op1" value="0"><caption>Option 1</caption></enum>
<!-- The value of an enum is equal to the name attribute value-->
<enum name="op2" value="1"><caption>Option 2</caption></enum>
</node>
<node name="field10" class="FreeEnum" default="Option 1" enum="Option 1¦Option
2">
<!--Values are separated by ¦ -->
<shortdesc>Field 10</shortdesc>
<longdesc>Field 10's help</longdesc>
</node>
<node name="field11" class="Memo">
<shortdesc>Field 11</shortdesc>
<longdesc>Field 11's help</longdesc>
<default>this is a multiline default value</default>
```

```
<!-- In the case of a Memo, the default value is set in a default node.-->
 </node>
</node>
```

設定ページ

defaultノードとadvancedノードは、<layout>要素を統括しています。この要素は、Javaコネクタの設定ウィザードの［コネクタの導入］ページのレイアウトを定義します。

ページの内容は枠にまとめられることもあります。各枠は同種のプロパティをまとめます。例：接続プロパティ用の枠、など。

各<frame>要素は、

- 名前で定義されます。
- 枠に説明を加える<caption>要素を含みます。

各<frame>要素は、表示する設定プロパティの参照を含んでいます。

設定プロパティが枠で参照されていない場合は、プロパティの値をウィザード内でパラメータ設定することはできません。高度な設定モードのみでプロパティが表示される場合は、プロパティをadvancedノード内で参照します（defaultノードではありません）。

例

```
<node name="default" class="ConWiz">
   <layout>
    <frame name="frame1">
     <caption>Frame #1</caption>
     <attribute name="field1"/>
     <attribute name="field2"/>
     <attribute name="field5"/>
    </frame>
    <frame name="frame2">
     <caption>Frame #2</caption>
     <attribute name="field6"/>
    </frame>
    <frame name="frame3">
     <caption>Frame #3</caption>
     <attribute name="field3"/>
     <attribute name="field4"/>
    </frame>
   </layout>
  </node>
```

DSD - 頻発する問題点

以下の問題点はDSDファイルを無効にします。

- 終了タグの欠如
- 無効な属性値
- 「name」属性がタグの第1属性として宣言されていない。
- **ルート**ノードが指定されていない。
- 不明なクラスが使用されている。
- defautまたはadvancedノードが不在である。または、defautまたはadvancedノードは**ルート**ノードから継承しない。
- 不明な属性が参照されている。
- ルートノードが定義するプロパティは、導入ファイル（ra.xml）内に存在しない。

A 問題点の報告方法

付録

弊社製品の品質を常に改善するためには、ユーザであるお客様に、製品の使用時に発見された問題点をペレグリンシステムズの顧客サポートに報告していただくことが非常に大切です。問題点が迅速に識別され解決されるためには、問題点に関する詳細な情報が必要になります。しかし弊社に提供される情報が不完全なために、問題を再現できないことも多々あり、よって、問題点の解決も不可能なことがあります。

本章では、製品の問題点をより適切に顧客サービスに報告するために、どの情報が不可欠であるかを説明します。

一般的な情報

顧客サービスに必ず報告しなければならない情報は以下の通りです。

- 機能に関する情報 [p. 253]
- 技術的な情報 [p. 254]

機能に関する情報

以下の情報により、問題点の分類が可能になります。

- お客様の会社名

- 問題が発生する製品名
- 問題が発生する機能･場所（例えば、Connect-Itのシナリオビルダ、AssetCenterの調達管理、など）
- 問題が発生する頻度（問題は毎回発生するか、または問題の発生はある一定の要因やデータにより左右されるのか、など）

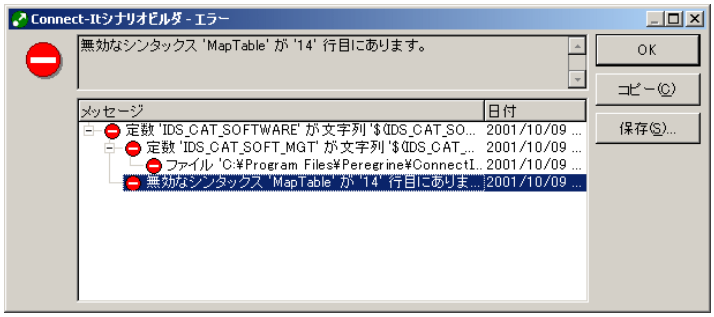
技術的な情報

重要な技術的情報の一部は、以下の手順に従って製品内で得ることができます。

1 **［ヘルプ／バージョン情報］**メニューを選択します。
2 表示されるダイアログボックス内で**［詳細］**をクリックします。システムと製品に関する技術的な情報が表示されます。
3 **［コピー］**ボタンをクリックします。情報はWindowsのクリップボードにコピーされます。
4 新規ファイルにこの情報を貼り付け、ペレグリンシステムズの顧客サポートまで御送付ください。

問題発生時にアプリケーション内でエラーメッセージが表示される場合は、以下の手順に従って問題点を報告してください。

1 エラーメッセージが発生したら**［コピー］**ボタンをクリックします。これにより、情報はWindowsのクリップボード内にコピーされます。



注意:

上記の画面は例に過ぎません。エラーメッセージの内容やソフトウェア名は、発生する問題により異なります。

2 新規ファイルにこの情報を貼り付け、ペレグリンシステムズの顧客サポートまで御送付ください。

問題固有の情報

弊社のサポートが問題を再現するためには、先ず、問題発生に至るまでの状況と過程の詳細なリストが必要になります。問題の発生は様々な要因に左右されるため、例えば以下の様に詳細に説明してください。

1 資産「Peregrine Prosigma 850」をAssetCenterの資産のテーブルで選択します。この資産は、製品付属のデモ用データベースに含まれているものです。
2 この資産の詳細画面で［取得］タブをクリックします。
3 ［取得］タブの［調達］サブタブページをクリックします。
4 ［市場価格］フィールド値を123.45に変更します。
5 ［変更］をクリックします。
6 ...

ヒント:

原則として、問題発生の状況はできる限り詳細に記述してください。

その他の情報

問題のより適切な識別のために、スクリーンショットやその他のファイルをサポートに送付することも可能です。

注意:

Eメールで大型サイズのファイルを送るのはお控えください。添付ファイルのサイズは500 KBまでに制限してください。必要であれば、ペレグリンのサポートは、500 KB以上のファイルを受信するためにFTPサイトを開くことも可能です。

送信するファイルのサイズを縮小するには、

- 圧縮ツールを使用します。多くの場合、ファイルのサイズが大幅に縮小されます。
- 画像の場合は、色数を16に制限します。これで画像のサイズは大幅に縮小されます。

Connect-Itに関する問題を報告する場合

Connect-Itはドキュメントログ、アプリケーションログと、サービスログにデータを常時記録しています。これらのファイル内の情報は、問題の再現に非常に有用です。

これらのファイルは以下のフォルダに格納されています。

- bin32\conitgui.log：アプリケーションログ
- bin32\<サービス名>.log：Connect-Itサービスログ
- 「.idx」、「.dat」と「.msg」ファイル：シナリオに関連するログファイル。ファイル名は、Connect-Itのグラフィカルインタフェース内のシナリオを使って設定できます。

注意:

Connect-Itの問題点を報告する際には、上記のファイルとシナリオファイル（「.scn」）をサポートまで御送付ください。

B 用語解説

付録

この用語解説では、Connect-Itで使用される主要用語の一部を説明します。また、データベース言語、Basicスクリプト、Java、外部アプリケーションで使用される主要用語も列挙されています。

Connect-It用語

コネクタ

コネクタにより、Connect-Itは、外部アプリケーションからまたは外部アプリケーションへデータを処理することができます。コネクタ間でデータを交換するために、コネクタはデータをドキュメントに転記し、ドキュメントの生成または取り込みを実行します。

コネクタは数種に分類されます。

- インベントリコネクタ

  インベントリコネクタは、企業の資産、特にIT資産をスキャン（走査）するデータベースのデータを処理します。コネクタにはIntel LanDeskコネクタ、SMS 1.xコネクタとSMS 2.xコネクタなどがあります。
- Peregrine Systemsコネクタ

Peregrine Systemsコネクタは、ペレグリンシステムズアプリケーション用のコネクタです。Asset Managementコネクタ、ServiceCenterコネクタ、InfraTools Managementコネクタ、InfraTools Desktop Discoveryコネクタ、Action Request Systemコネクタなどがあります。

- アプリケーションコネクタ

  アプリケーションコネクタは、外部のデータアプリケーション用のコネクタです。Lotus Notesコネクタ、NTセキュリティコネクタなどがあります。
- プロトコルコネクタ

  プロトコルコネクタは、特定のプロトコルを用いて外部アプリケーションによりフォーマットされたデータを処理します。XMLコネクタ、テキストコネクタ、データベースコネクタがあります。
- Javaコネクタ

  JavaコネクタはJavaで開発されています。Connect-ItのJava開発キットを使用すると、ユーザは独自のコネクタを開発できます。

  Javaコネクタ作成の詳細については、「Connect-ItのJava開発キット（JDK）[p. 211]」を参照してください。

コネクタの役割は以下の通りです。

- コネクタは、コネクタの接続先のアプリケーションで使用可能なデータ集合に対応する、使用可能なドキュメントタイプを発行します。

  例：

  Eメールコネクタ（受信）が使用可能なドキュメントタイプを発行する場合、この使用可能なドキュメントタイプの内容は、From（送信者）構造体、Carbon Copy（CC）コレクションとAttachment（添付ファイル）コレクションなど、Eメール内の全データに相当します。

- ドキュメントの生成と取り込み

  ドキュメントを生成するために、コネクタは、ある特定の統合シナリオ用に作成された「生成用ドキュメントタイプ」テンプレートを使用します。コネクタはこのテンプレートに基づいてドキュメントを生成します。

  例：

  付属のInfraTools Desktop Discovery - Asset Management各シナリオで、Desktop Discoveryコネクタは、「Machine」という生成用ドキュメントタイプをテンプレートとして使用してドキュメントを生成します。この生成用ドキュメントタイプは、InfraTools Desktop Discoveryアプリケーションが IT資産をスキャンする際に取得する全データの部分集合に当たります。

ドキュメントを取り込むために、コネクタは、マッピングボックスから送られてくるドキュメントの値を抽出し、外部アプリケーション用にデータ値を転換します。

データが、外部ソースアプリケーションから外部ターゲットアプリケーションに転送される場合、コネクタは、ソースコネクタとターゲットコネクタと呼ばれます。ソースコネクタはドキュメントを生成し、ターゲットコネクタはドキュメントを取り込みます。

ドキュメントとドキュメントタイプ

ドキュメントはConnect-It内でデータを伝達する媒体です。各ドキュメントは、外部アプリケーションの1つのデータ集合に対応します。外部アプリケーションの機能に応じて、1データ集合（データコンテナ）は以下の内容になります。

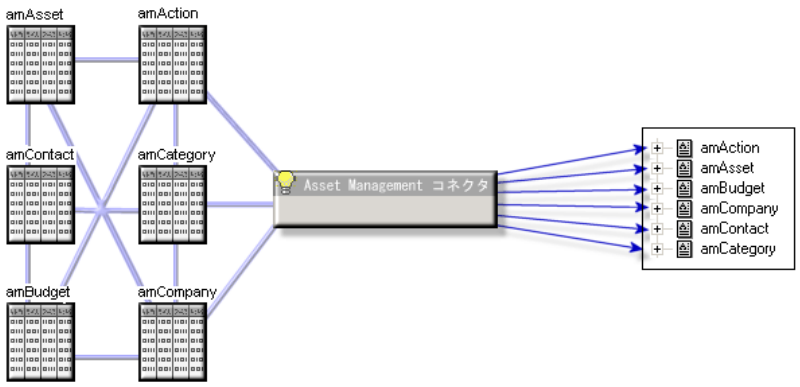
- 1つのデータベーステーブル
- 1通のEメールメッセージ
- インベントリ情報を含む1つのファイル
- 区切り文字で区切られた1つのテキストファイル
- 1つのXMLファイル
- セキュリティ情報
- その他

シナリオビルダでコネクタが開かれると、コネクタは使用可能な全てのドキュメントタイプのリストを発行します。使用可能な全ドキュメントタイプのリストは、外部アプリケーションで使用可能なデータ集合全体を指します。

例：

AssetCenterデータベースの全テーブルは、シナリオビルダ内のAsset Managementコネクタが発行する全ての使用可能なドキュメントタイプ群に一致します。

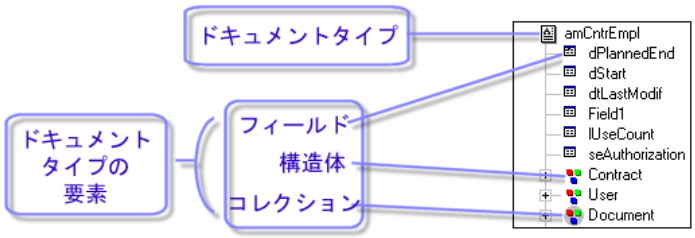
図 B.1. Asset Managementコネクタ - 使用可能なドキュメントタイプ群



ドキュメントタイプの要素

各ドキュメントタイプはツリー構造になっています。ツリー構造内には、単純な自立型要素（フィールド）と、複数レベルの構造を持つ複雑な要素があります。後者には更に、ルートノードや、単純または複雑な要素を含む構造体やコレクションが含まれています。

図 B.2. ドキュメントタイプの要素



単純要素には以下のものがあります。

- フィールド（）

  フィールドは、数、倍長整数、固定長テキスト、日付、などの特定のデータ型を含みます。
- BLOB型フィールド（）

  BLOB（binary large object）型フィールドは、画像、音声、ビデオなどの保存される必要のあるバイナリオブジェクトを含んでいます。
- メモ型フィールド（）

  メモ型フィールドは可変な長さのテキストを含んでいます。

複雑な要素には以下のものがあります。

- 単純要素または複雑要素を含む構造体（）

  例1：データベーステーブル用のドキュメントタイプでは、各構造体は別のテーブルを指しています。構造体は、2つのテーブル間のリンクが1対1であることを示しています。つまり1番目のテーブルのレコード1つは、別のテーブルのレコード1つにのみリンクされています。

  例2：受信されたEメールメッセージ用のドキュメントタイプでは、構造体はメッセージの送信者に関する情報に当たります（1つのメッセージには1人の送信人しかいません）。
- 単純要素または複雑要素を含むコレクション（）

  例1：データベーステーブル用のドキュメントタイプ内にコレクションがある場合、このコレクションはデータベース内の別のテーブルを指しています。コレクションでは、2つのテーブル間のリンクは1対N、またはN対Nです。1番目のテーブルの1つのレコードは、2番目のテーブルの1つまたは複数のレコードにリンクされており、2番目のテーブルのレコードも、1番目のテーブルの1つまたは複数のレコードにリンクされています。

  例2：送信されたEメールメッセージ用のドキュメントタイプでは、コレクションはメッセージの受信者に当たります（1つのメッセージの受信者数は、何人にでもなり得ます）。

注意:

属性のコレクションは、1つのフィールドのみを含むコレクションです。フィールドは属性に当たります。コレクション内の各構成要素は、1つの要素がN回繰り返された構造体です。

コネクタがドキュメントを処理する前に、ドキュメント内の各コレクションには不定数の構成要素があります。これらの構成要素の正確な数を処理前に知ることはできません。そのため、コレクションはN個の構成要素から成っている、と定義します。

図 B.3. ドキュメントタイプのコレクション

例 :

Desktop Discoveryコネクタが発行する使用可能なドキュメントタイプ「Machine」では、コンピュータにインストールされたソフトウェアはコレクションで示されています。これは、各コンピュータにインストールされているソフトウェアの数は不定なため、この種のデータはコレクションの形を取る必要があるからです。コレクションの各構成要素はソフトウェアの内容を説明しています（名前、ソフトウェア会社、バージョンなど）。

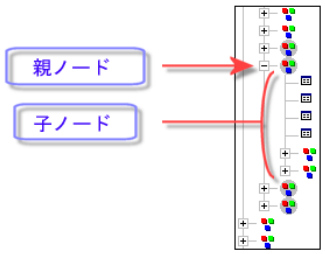
ドキュメントタイプ内の親ノードと子ノード

ツリー構造内では、階層を持つ要素下には子要素があります。ドキュメントタイプでは、これを親ノードと子ノードと呼びます。

注意:

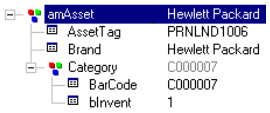
親ノードと子ノードの概念はドキュメントの処理中に重要です。例えば、子ノード内のフィールドが拒否された結果、親ノードも拒否されることがあります。

図 B.4. ドキュメントタイプのツリー構造



生成用ドキュメントタイプと取り込み用ドキュメントタイプ

コネクタがドキュメントを生成し取り込むために、コネクタは、生成用ドキュメントタイプと取り込み用ドキュメントタイプを使用する必要があります。生成用ドキュメントタイプ、または取り込み用ドキュメントタイプを作成する場合、Connect-Itのユーザは、コネクタが発行する使用可能なドキュメントタイプから要素を選択します。例えば、データベース型コネクタの生成用ドキュメントタイプには、外部ターゲットアプリケーションにインポートする必要のあるフィールドのみを選択します。

使用可能なドキュメントタイプ	生成用または取り込み用ドキュメントタイプ	生成されるドキュメント
		
コネクタは使用可能なドキュメントタイプを発行します。上記の図は、Asset Managementコネクタに発行される使用可能なドキュメントタイプ「amAsset」の抜粋です。	生成用、または取り込み用ドキュメントタイプを作成するには、使用可能なドキュメントタイプから必要な要素を選択します。	コネクタは、生成用または取り込み用ドキュメントタイプに沿って、ドキュメントの生成または取り込みを実行します。

XMLファイルとDTDファイル

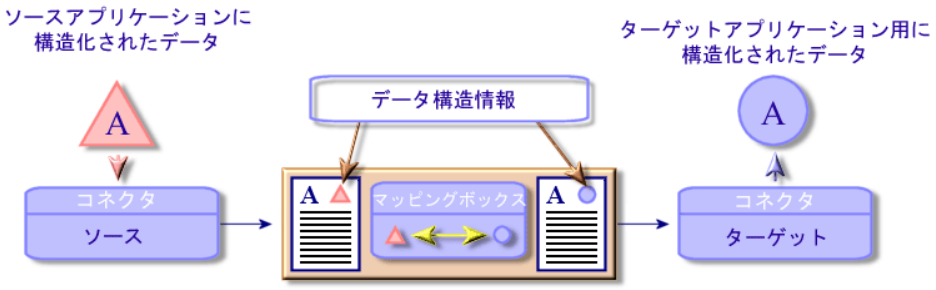
Connect-It内で使用されるドキュメントはXML（Extensive Markup Language）ファイルに相当し、ドキュメントが使用するドキュメントタイプは、DTD（Document Type Definition、文書型定義）に相当します。

Connect-Itのシナリオビルダでは、全てのドキュメントをXMLファイルに転換し、全てのドキュメントタイプをDTDに変換することができます。

マッピング

マッピングは、ソースコネクタの生成用ドキュメントタイプの要素群と、ターゲットコネクタの取り込み用ドキュメントタイプの要素群を一致させ、データを変換します。

Connect-Itのマッピングボックスにより、ターゲットコネクタは、ソースコネクタが生成したドキュメントを取り込むことができます。マッピングボックスは内部ツールで、他のコネクタのようにドキュメントの生成や取り込みを行いますが、外部アプリケーションには接続しません。



統合シナリオ

統合シナリオにより、異種のアプリケーション間でデータを伝達できるようになっています。

シナリオを作成するには以下の手順に従います。

1 ソースコネクタとターゲットコネクタを選択します。

原則として、使用するコネクタの数に制限はありません。

2 コネクタを設定します。

3 コネクタをリンクします。
4 ソースコネクタでは、生成用ドキュメントタイプを作成し、ターゲットコネクタでは取り込み用ドキュメントタイプを作成します。
5 ソースドキュメントタイプとターゲットドキュメントタイプ間のマッピングを作成します。
6 スケジュールを選択します。

スケジュールは、シナリオのソースコネクタがドキュメントを生成する頻度を設定します。

Connect-Itは用例シナリオと共に提供されています。

警告:

用例シナリオは、多くの場合ユーザの特定の事例には当てはまりません。

ルール（ディレクティブ）

ルール（ディレクティブ）は、Connect-Itのコンポーネントがドキュメントを生成し、取り込むために使用する情報のことです。これには**生成用ルール**と**取り込み用ルール**があります。

ドキュメントログ

シナリオビルダ内のタブにあるドキュメントログでは、Connect-Itのコネクタやその他のコンポーネント（リンクやマッピングボックスなど）が生成した、または取り込んだ全ドキュメントの内容を確認できます。

コネクタがドキュメントを拒否すると、ドキュメントログは拒否されたドキュメントをアイコンと共に表示します。ドキュメントログは更に、コネクタまたはマッピングボックスが不完全にドキュメントを生成したり、取り込んだりした場合にも警告を発します。

スケジュール

Connect-Itのスケジュールでは、統合シナリオのソースコネクタがいつドキュメントを生成するのかを設定できます。スケジュールを作成するには、ドキュメントを1つまたは複数のスケジューラに関連付けます。

スケジューラはタイマの役割を果たしており、コネクタが特定の日時に（例：2002年10月4日）、または定期的に（例：一定の期間内で2時間おき）ドキュメントを生成するように設定します。

処理レポート

処理レポートは、コネクタやマッピングボックスがドキュメントを処理するたびに生成するドキュメントのことです。このレポートによりドキュメントの処理に成功したか、または問題が起こったかどうかを確認できます。更に、別のコネクタがこの処理レポートを取り込むと、Connect-Itで発生した問題について警告を出すようにも設定できます。

例：

Eメールコネクタはシナリオの処理レポートを取り込み、メッセージの形でConnect-Itの管理者へ処理レポートを送信することができます。

認証証明書

認証証明書は、ベースコネクタと、選択したConnect-Itパッケージに含まれているオプションコネクタと追加コネクタを、使用可能にするテキストファイルです。

外部アプリケーション

外部アプリケーションは、コネクタの接続先であるアプリケーションやデータソースを指します。例えば、AssetCenterアプリケーション、XMLファイル、メッセージシステムなどです。

Blob

Blob（Binary Large Object）タイプのフィールドはバイナリデータ（サウンド、ビデオ、画像ファイル）のフィールドです。

キャッシュ

コネクタが使用するキャッシュファイルは、コネクタが発行する使用可能なドキュメントタイプの記述を含みます。キャッシュファイルの拡張子は「CCH」です。キャッシュを使用するとドキュメントタイプの記述を取得する必要がなくなるため、コネクタを速く使用できます。

コレクション

ドキュメントタイプ内で、コレクションは1つまたは複数の要素と、別の複数の要素間の関係を表しています。

例：データベーステーブル用のドキュメントタイプ内にコレクションがある場合、このコレクションはデータベース内の別のテーブルを指しています。コレクションでは、2つのテーブル間のリンクは1対N、またはN対Nです。1番目のテーブルの1つのレコードは、2番目のテーブルの1つまたは複数のレコードにリンクされており、2番目のテーブルのレコードも、1番目のテーブルの1つまたは複数のレコードにリンクされています。

例：送信されたEメールメッセージ用のドキュメントタイプでは、コレクションはメッセージの受信者に当たります（1つのメッセージの受信者数は、何人にでもなり得ます）。

コンポーネント

コンポーネントは、コネクタまたは内部ツール（マッピングボックス）に当たります。

サービスコンソール

グラフィカルインタフェースであるConnect-Itコンソールでは、シナリオを管理し、シナリオに関連付けるサービスを作成することができます。

取り込み

取り込みとは、ターゲットコネクタが外部アプリケーションにドキュメントを書き込んだり送信したりするプロセスを指します。

DAT

拡張子「DAT」を含むファイルは、シナリオに処理されるドキュメントに相当します。

これらのファイルは、ドキュメントログの設定用ウィンドウ（**［ログ／ドキュメントログを設定する］**）で、**［ファイルを使用］**オプションを選択した場合にのみ生成されます。

DBK

DBKファイルは、シナリオ文書に対応するXMLファイルです。これらのファイルはDocBook DTDに準拠しています。DBKファイルの詳細については、「シナリオ文書 [p. 155]」の章を参照してください。

ルール（ディレクティブ）

コネクタのルール（ディレクティブ）は、コネクタがドキュメントの生成または取り込み時に従う指示を指します。

例：データベース型のコネクタでは、ソースデータベースにフィルタを適用するWHERE句とORDERBY句は、取り込み用ルールに当たります。

ルールの詳細については、マニュアル『**コネクタ**』の「**コネクタのルール（ディレクティブ）**」の章を参照してください。

コネクタの導入

Javaコネクタをシナリオビルダ内に導入するためには、以下の内容を指定します。

- 導入ファイル
- コネクタがドキュメントを生成するか、または取り込むかどうか
- ツールボックス内に表示されるコネクタ用のアイコン
- 数種のオプション

コネクタの導入の詳細については、「Connect-ItのJava開発キット（JDK） [p. 211]」の章の「イベントコネクタを作成する [p. 240]」の節、「Javaコネクタの導入 - イベントコネクタ [p. 242]」を参照してください。

シナリオビルダ

シナリオビルダは、統合シナリオ作成用のグラフィカルインタフェースです。シナリオは、ファイルの拡張子「SCN」で識別されます。

要素

ドキュメントタイプのツリー構造には以下の要素が含まれます。

- 構造体
- コレクション
- フィールド

この要素は、ドキュメントタイプの端末要素に当たります。

ユーザフォーマット

ユーザフォーマットとは、ユーザがマッピングスクリプト内で使用するために定義する日付型または数値のフォーマットを指します。フォーマットは、**PifUserFmtVarToStr**関数と**PifUserFmtStrToVar**関数と共にのみ使用されます。使用に関する詳細は、オンラインの『プログラム用参考ガイド』（Connect-Itでのスクリプト作成中に「F1」キーを押すと表示されます）を参照してください。

ユーザフォーマットの詳細については、「マッピングスクリプト [p. 109]」の章の「ユーザフォーマット [p. 137]」の節を参照してください。

IDX

IDXファイルは、ドキュメントログ内のDATファイルとLOGファイルのデータを、迅速に検索するためのインデックスファイルです。

これらのファイルは、ドキュメントログの設定用ウィンドウ（**［ログ／ドキュメントログを設定する］**）で、**［ファイルを使用］**オプションを選択した場合にのみ生成されます。

MSG

MSGファイルは、ドキュメントログ内にあるメッセージを含みます。

これらのファイルは、ドキュメントログの設定用ウィンドウ（**［ログ／ドキュメントログを設定する］**）で、**［ファイルを使用］**オプションを選択した場合にのみ生成されます。

スケジュールのポインタ

スケジュールのポインタを使用すると、シナリオが処理するデータの量を少なくすることができます。

ポインタがレコードの最終変更日である場合、コネクタは最後の起動以降に作成または更新されたレコードのみを処理します。

シナリオがスケジュールモードで起動していると、生成用ドキュメントタイプのポインタのステータスにより、コネクタは前回のセッションで処理されなかったデータのみを処理できます。

例：Asset Managementコネクタは、最後のセッションで9時00分00秒に資産のテーブルのレコードを読み取ったとします。次のセッションでは、コネクタは9時00分01秒から作成されたレコード全てを読み取ります。

生成

生成とは、データをドキュメントに転換するために、ソースコネクタが外部アプリケーションからデータを読み取ったり受信したりするプロセスのことを指します。

スケジューラ

スケジューラは、シナリオのソースコネクタの1つまたは複数の生成用ドキュメントタイプに関連付けられたタイマです。

シナリオのスケジューラを編集するには、**［シナリオ／スケジューラ］**を選択します。

シナリオビルダには2つの既製スケジューラが付属しています。

- **「一回」**スケジューラ

  このスケジューラは、シナリオのすべての生成用ドキュメントタイプにデフォルトで関連付けられています。
- **「同期」**スケジューラ

  このスケジューラにより、コネクタは毎秒ドキュメントを生成します。

実行結果

各セッション後に、シナリオビルダは、シナリオの各コンポーネントが処理したドキュメントの数に関する実行結果を取得します。例えば、Asset Managementコネクタが1セッション中に処理したドキュメントの数などです。

各コネクタの実行結果を表示するには、シナリオ図内のコネクタの上部にポインタを置きます。

構造体

ドキュメントタイプ内の構造体は、この構造体を含むノードと1対1の関係であることを表しています。構造体を含むノードは、ドキュメントタイプのルートノード、構造体またはコレクションです。

例：データベーステーブル用のドキュメントタイプでは、各構造体は別のテーブルを指しています。構造体は、2つのテーブル間のリンクが1対1であることを

示しています。つまり1番目のテーブルのレコード1つは、別のテーブルのレコード1つにのみリンクされています。

処理後のアクション

一部のコネクタ（InfraTools Desktop Discoveryコネクタなど）は、自分が生成したドキュメントに更に操作を加えることができます。この場合、コネクタの設定ウィザードのページで、生成されるドキュメントの「処理後のアクション」を選択します。例えば、Connect-Itで処理に成功した後にソースデータを削除するなどのアクションがあります。

ピボットドキュメントタイプ

ピボットドキュメントタイプは、構成要素（資産、ソフトウェア、従業員など）の一般的な表現に当たります。ソースコネクタは、自分のドキュメントタイプとピボットドキュメントタイプを一致させます。

ピボットドキュメントタイプは、マッピングなしで、ソースコネクタとターゲットコネクタ間でデータを転送できるようにします。

ログビューア

ログビューアでは、Windowsのサービスとしてシナリオが起動するたびに作成されるLOGファイルを読むことができます。

ビュー

シナリオ図のビューは、シナリオ図を見やすくするために使用されます。例えば、多数のコンポーネントを含むシナリオでは、1つまたは一部のコンポーネントのみを表示するビューを作成できます。

ビューの詳細については、「シナリオビルダ [p. 27]」の章の「メインウィンドウ [p. 27]」の節、「シナリオ図のビュー [p. 33]」を参照してください。

DSC

書式設定ファイル（ファイルの拡張子「.dsc」）は、テキストファイル内のデータがどのように構成されているかを説明するファイルです。ファイル内のデータは、データベースのテーブル内のフィールド値に当たります。

テキストコネクタの使用可能なドキュメントタイプは、書式設定ファイル内で作成されたドキュメントタイプです。

例：ある書式設定ファイルでは以下の内容を指定します。

- テキストファイルは1つのテーブル（従業員のテーブル）のデータを含みます。
- テキストファイルの1行目では、列のタイトルを指定します（各列はテーブルのフィールド1つに対応します）。
- 値は「~」で区切ります。

主要用語

AQL

AQL（Advanced Query Language）は、AssetCenterデータベースのデータにアクセスするために、AssetCenterが使用する言語です。これはSQLに匹敵します。クエリの実行時に、AQLはデータベースエンジンのSQL言語に自動的に変換されます。

DSE

DSE（Directory Entry Service）は、LDAPディレクトリのツリー構造を構成するエントリです。これらの情報は、ツリー形式で表現されます。

LDAPディレクトリの各エントリは、抽象オブジェクトまたは実際のオブジェクト（例えば、人、物品、パラメータなど）に対応します。

FSF

拡張子が「.fsf」（Fingerprint Save File）であるファイルは、あるコンピュータに関する全データを含みます。

「.fsf」ファイルは、InfraTools Desktop Discovery型のスキャンで生成されます。スキャン時に収集され「.fsf」ファイルに格納された情報は、コンピュータに関するデータを更新するために（整合性チェックの段階などで）分析されます。

JCA

Sun MicrosystemsのJCA（Java Connector Architecture)は、インタフェースを説明する仕様全体をまとめたものです。

JCAの主な目的は、異なる技術に基づいたアプリケーションとJavaサーバが対話できるようにする規格の層を定義することです。

JCAは、接続インタフェースや標準コネクタの調整を簡易化します。

整合性チェック

整合性チェックとは、別アプリケーションから受信するデータの統合であり、ソースアプリケーションから来るデータの方が、ターゲットアプリケーション内の既存のデータよりも新しいと見なされています。

- 既存しない場合データは挿入されます。
- 既存する場合、受信するデータの新規情報に応じて既存のデータは更新されます。

このプロセスは、受信するデータが、ターゲットアプリケーション内に既存するかどうかに応じて実行されます。

ターゲットアプリケーションが空である場合は、このプロセスは整合性チェックではなく「インポート」になります。

XMLスキーマ

XMLスキーマとは、XMLドキュメント内で使用される内容の定義です。XMLスキーマは、SGML規格のスキーマであるDTDのスーパーセット（上位集合）に当たります。

DTDとは違って、XMLスキーマはXMLシンタックスで書かれています。このシンタックスはDTDよりも拡張されています。

XMLスキーマは、全XMLツールで作成可能です。

マップテーブル

マップテーブルは、ある集合の要素を別の集合の要素に対応させる表です。

トランザクション

トランザクションとは、コンピュータとユーザ間または2台のコンピュータ間の対話における1要素です。例えば、情報の要求とそれに対する応答などを指します。

グローバル変数

グローバル変数の値は、変数を定義するモジュールだけでなく、プログラムの全命令がアクセスでき、変更できる値です。

XSL

XSL（eXtensible StyleSheet Language）は、XML専用に開発された拡張可能なスタイルシート言語です。これは公式な規格とは見なされていません。

索引

目次

P

R

S

T

U

W

X


Peregrine
SYSTEMS