

**IBM WebSphere Business Integration
Adapters**



Adapter for PeopleSoft ユーザーズ・ガイド

バージョン 3.0.x

**IBM WebSphere Business Integration
Adapters**



Adapter for PeopleSoft ユーザーズ・ガイド

バージョン 3.0.x

お願い

本書および本書で紹介する製品をご使用になる前に、129 ページの『付録 C. 特記事項』に記載されている情報をお読みください。

本書は、WebSphere Business Integration Adapter for PeopleSoft (5724-G98) バージョン 3.0.x に適用されます。
本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。
<http://www.ibm.com/jp/manuals/main/mail.html>

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは
<http://www.ibm.com/jp/manuals/> の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原 典： IBM WebSphere Business Integration Adapters
Adapter for PeopleSoft User Guide
Version 3.0.x

発 行： 日本アイ・ピー・エム株式会社

担 当： ナショナル・ランゲージ・サポート

第1刷 2004.7

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財)日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

© Copyright International Business Machines Corporation 2001, 2004. All rights reserved.

© Copyright IBM Japan 2004

目次

本書について	v
対象読者	v
関連文書	v
表記上の規則	vi
本リリースの新機能	vii
リリース 3.0.x の新機能	vii
リリース 2.3.x の新機能	vii
リリース 2.2.x の新機能	vii
リリース 2.1.x の新機能	viii
リリース 2.0.x の新機能	viii
リリース 1.4.x の新機能	ix
リリース 1.3.x の新機能	ix
リリース 1.2.x の新機能	ix
リリース 1.1.x の新機能	x
第 1 章 概要	1
用語	1
コネクタ・コンポーネント	3
コネクタの動作方法	4
第 2 章 コネクタのインストール	13
アダプター環境	13
前提条件	14
アダプターと関連ファイルのインストール	16
インストールの検査	17
コネクタ用のアプリケーションを使用可能にする	18
第 3 章 コネクタの構成	27
Connector Configurator の概要	27
Connector Configurator の始動	28
System Manager からの Configurator の実行	29
コネクタ固有のプロパティ・テンプレートの作成	29
新しい構成ファイルを作成	32
既存ファイルの使用	33
構成ファイルの完成	35
構成ファイル・プロパティの設定	35
構成ファイルの保管	42
構成ファイルの変更	43
構成の完了	43
グローバル化環境における Connector Configurator の使用	44
第 4 章 コネクタ用のビジネス・オブジェクトについて	45
ビジネス・オブジェクトおよび属性の命名規則	45
ビジネス・オブジェクトの構造	46
ビジネス・オブジェクトの作成	51
ビジネス・オブジェクト動詞の処理	55
ビジネス・オブジェクトの属性プロパティ	64
ビジネス・オブジェクトのアプリケーション固有情報	66
第 5 章 PeopleSoftODA を使用したビジネス・オブジェクト定義の生成	73

インストールと使用方法	73
Business Object Designer での PeopleSoftODA の使用	77
生成された定義の内容	85
ビジネス・オブジェクト定義ファイルのサンプル	88
BO_PsftEmployee ビジネス・オブジェクト	89
SavePostChange ビジネス・オブジェクトの例	94
ビジネス・オブジェクト定義への情報の追加	95
第 6 章 コネクターの実行	97
コネクターの始動	97
コネクターの停止	98
コネクターのアップグレード	99
第 7 章 トラブルシューティングとエラー処理	101
始動時の問題	101
始動時の問題 (WebSphere InterChange Server ブローカーのみ)	101
始動時の問題 (WebSphere MQ Integrator Broker のみ)	102
処理時の問題	102
マッピング (ICS 統合ブローカーのみ)	103
エラー処理とロギング	103
アプリケーションへの接続の切断	104
付録 A. コネクターの標準構成プロパティ	105
新規プロパティと削除されたプロパティ	105
標準コネクタ・プロパティの構成	105
標準プロパティの要約	107
標準構成プロパティ	111
付録 B. コネクタ固有のプロパティ	125
AppServerMachineNameOrIP	125
ApplicationPassword	125
ApplicationUserName	125
ConnectorID	126
ConvertToPrimitiveFloat	126
DisableCrossReferencing	126
EventKeyDelimiter	126
PIngCompInterface	126
PollQuantity	126
PortNumber	127
ReconnectSessionOnGetFail	127
SessionPoolSize	127
SetLangCode	127
PollFutureEvents	127
付録 C. 特記事項	129
プログラミング・インターフェース情報	131
商標	131

本書について

IBM^(R) WebSphere^(R) Business Integration Adapter ポートフォリオは、主要な e-business テクノロジー、エンタープライズ・アプリケーション、レガシー、およびメインフレーム・システムに統合コネクティビティを提供します。本製品には、コンポーネントをカスタマイズ、作成、および管理するためのツールとテンプレートが含まれており、これにより、ビジネス・プロセスの統合を実現します。

本書では、IBM WebSphere Business Integration Adapter for PeopleSoft のインストール、構成、ビジネス・オブジェクト開発、およびトラブルシューティングについて説明します。

対象読者

本書は、IBM のコンサルタントおよびお客様を対象としています。PeopleSoft および WebSphere Business Integration システム・アダプターの作成について、理解している必要があります。

関連文書

この製品に付属する資料の完全セットで、すべての WebSphere Business Integration Adapters のインストールに共通な機能とコンポーネントについて説明します。また、特定のコンポーネントに関する参考資料も含まれています。

本資料では、「システム・インストール・ガイド (Windows 版)」または「システム・インストール・ガイド (UNIX 版)」および「WebSphere InterChange Server システム・インプリメンテーション・ガイド」への参照が数多く含まれています。本書を印刷する場合は、これらの資料も印刷しておくといでしょう。

以下のサイトから、資料をインストールすることができます。

- アダプターの一般情報が必要な場合、アダプターを WebSphere Message Broker (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) とともに使用する場合は、およびアダプターを WebSphere Application Server とともに使用する場合は、以下のサイトを参照してください。

<http://www.ibm.com/websphere/integration/wbiadapters/infocenter>

- アダプターを InterChange Server とともに使用する場合は、以下のサイトを参照してください。

<http://www.ibm.com/websphere/integration/wicsserver/infocenter>

<http://www.ibm.com/websphere/integration/wbicollaborations/infocenter>

- Message Broker (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) の詳細については、以下のサイトを参照してください。

<http://www.ibm.com/software/integration/mqfamily/library/manualsa/>

- WebSphere Application Server の詳細については、以下を参照してください。

<http://www.ibm.com/software/webservers/appserv/library.html>

上記のサイトには資料のダウンロード、インストール、および表示に関する簡単な説明が記載されています。

表記上の規則

本書は下記の規則に従って編集されています。

Courier フォント	コマンド名、ファイル名、入力情報、システムが画面に出力した情報など、リテラル値を示します。
太字	初出語を示します。
イタリック、イタリック 青い文字	変数名または相互参照を示します。 オンラインで表示したときにのみ見られる青の部分は、相互参照用のハイパーリンクです。青い文字ストリングをクリックすることにより、参照先オブジェクトに飛ぶことができます。
{ }	構文の記述行の場合、中括弧 { } で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
[]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプションのパラメーターです。
...	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、option[,...] は、複数のオプションをコンマで区切って入力できることを示します。
< >	個々の名前の要素を互いに区別するために、不等号括弧 < > で囲みます。例えば <server_name><connector_name>tmp.log のようにします。

本リリースの新機能

リリース 3.0.x の新機能

今回のリリースで追加された新機能は、以下のとおりです。

- Adapter for PeopleSoft に新たに追加されたサポートにより、ビジネス・オブジェクト定義にリストされる順序で属性を処理できるようになりました。
- 将来の有効日付のイベントに対するポーリングが追加されました。
- バージョン 3.0.x より、Adapter for PeopleSoft は Solaris 7 ではサポートされなくなりました。したがって、このプラットフォーム・バージョンに関する言及がこのガイドから削除されました。

リリース 2.3.x の新機能

今回のリリースで追加された新機能は、以下のとおりです。

- 内容による検索要求の処理をサポートします。
この機能により、代替キーの値によるレコード検索が可能になりました。
- アダプターは BigDecimal タイプ・データをサポートします。
- コネクタを、インバウンド・オブジェクト処理時にマルチスレッドで実行できるよう構成することができます。コネクタをマルチスレッドで実行する場合、単一スレッドの順次実行ポーリング用に 1 つ、セッション・インスタンスが予約されます。
- PeopleTools 8.14 から 8.43 をサポートします。ただし、PeopleTools の一部のバージョンに固有の機能は、それらのバージョンでのみサポートされます。
- PeopleSoft API ライブラリーの呼び出しを使用して、有効期限項目を挿入する場合の制限を設定するパラメーターが追加されました。
- アウトバウンド・オブジェクトの削除要求をサポートします。
- アダプターのインストール情報は、本書から移動しました。この情報の新たな入手先については、第 2 章を参照してください。
- バージョン 4.3 以降の Adapter for PeopleSoft は Microsoft Windows NT ではサポートされなくなりました。

リリース 2.2.x の新機能

今回のリリースで追加された新機能は、以下のとおりです。

- Adapter for PeopleSoft は HP-UX でサポートされるようになりました。
- 単一の ODA インスタンスを複数回再使用して、同じアプリケーション・サーバーに接続できます。
トランザクション中にアプリケーション・サーバーへの接続が失われた場合、またはセッション・インスタンスが無効になった場合は、コネクタが終了します。構成可能なコネクタ終了サポートが追加されました。(PingCompInterface)

- このコネクタにより追加される新しい発効日の順序の属性により、シーケンス番号に基づいてデータが挿入されます。

リリース 2.1.x の新機能

2003 年 3 月に更新されました。「CrossWorlds」という名前は、現在ではシステム全体を表したり、コンポーネント名やツール名を修飾するためには使用されなくなりました。コンポーネント名およびツール名自体は、以前とほとんど変わりません。例えば、「CrossWorlds System Manager」は現在では「System Manager」となり、「CrossWorlds InterChange Server」は「WebSphere InterChange Server」となっています。

以下は、今回のリリースで追加された新機能です。

- PeopleSoft Adapter (PeopleSoft ODA を含む) は、PeopleTools の 8.18 から 8.40 のバージョンのみを完全にサポートします。
- XML の Application Designer プロジェクトが PeopleTools でサポートされるようになったため、前のバージョンのすべての CrossWorlds イベント通知オブジェクトは、更新するか、新しいフォーマットでインポートする必要があります。詳細については、PeopleSoft サポートを参照してください。
- ビジネス・オブジェクト ASI が拡張されて、PeopleSoft の新しいプロパティ `setEditHistoryItems` が組み込まれました。この値を `true` に設定すると、アダプターを修正モードで実行することができ、PeopleSoft の履歴・レコードを変更できます。ただし、この値が `getHistoryItems` プロパティに取り込まれる前、リリース 8.4 以降の PeopleTools では、この値は履歴項目を検索する `getHistoryItems` と、編集を可能にする `setEditHistoryItems` という 2 つの独立したプロパティに抽出されていました。これら 2 つのプロパティは、どちらもこのバージョンの Adapter for PeopleSoft でサポートされています。これらのプロパティの機能の詳細については、コンポーネント・インターフェースに関する PeopleSoft の資料を参照してください。
- PeopleSoft 内部のメッセージをアダプターのログ・ファイルに記録できるようになりました。現在のコンポーネント・インターフェースのコンテキストで PeopleCode 処理エラーまたは警告が発生すると、コネクタは `PSMessageCollection` オブジェクトからメッセージを検索して表示します。これらのメッセージの内容の詳細については、PeopleSoft コンポーネント・インターフェースの資料を参照してください。

リリース 2.0.x の新機能

IBM WebSphere Business Integration Adapter for PeopleSoft で提供されているコネクタは、国際化に対応しています。詳細については、11 ページの『ロケール依存型データの処理』および『付録 A. コネクタの標準構成プロパティ』を参照してください。

リリース 1.4.x の新機能

IBM WebSphere Business Integration Adapter for PeopleSoft には、PeopleSoft 用のコネクターが組み込まれています。このアダプターは、InterChange Server (ICS) と WebSphere MQ Integrator (WMQI) の両方の統合ブローカーと共に動作します。統合ブローカーとは、異種のアプリケーション・セット間の統合を実行するアプリケーションです。統合ブローカーは、データ・ルーティングなどのサービスを提供します。

このアダプターには、以下の要素が含まれます。

- PeopleSoft に固有なアプリケーション・コンポーネント。
- PeopleSoftODA for PeopleTools バージョン 8.16 以降。
- サンプル・ビジネス・オブジェクト。¥connectors¥PeopleSoft¥Samples¥ にあります。
- IBM WebSphere Adapter フレームワーク。コンポーネントは以下のとおりです。
 - コネクター・フレームワーク
 - 開発ツール (Business Object Designer と Connector Configurator を含む)
 - API (ODK、JCDK、および CDK を含む)

本書では、このアダプターを InterChange Server (ICS) と WebSphere MQ Integrator (WMQI) の両方の統合ブローカーと共に使用するための情報を提供します。

重要: コネクターは国際化に対応していないため、ISO Latin-1 データのみが処理されることが確実である場合を除いて、コネクターと InterChange Server バージョン 4.1.1 を併用しないでください。

リリース 1.3.x の新機能

重要: コネクターは国際化に対応していないため、ISO Latin-1 データのみが処理されることが確実である場合を除いて、コネクターと InterChange Server バージョン 4.1.1 を併用しないでください。

新規ツールの PeopleSoftODA は、コネクターのビジネス・オブジェクト定義を生成するために開発されました。詳細については、73 ページの『第 5 章 PeopleSoftODA を使用したビジネス・オブジェクト定義の生成』を参照してください。

リリース 1.2.x の新機能

このコネクター・バージョン 1.2.x 用のリリース資料には、次の新規または訂正情報が記載されています。

- 「In Status」(3) の状況となるイベントは、Init() の実行時にコネクターにより ReadyToPoll (0) にリセットされます。
- コネクター・プロパティの GetHistoryItems と setInteractiveMode は、除去されました。この機能は、現在では名前と値の対の形式で、ビジネス・オブジェクト・レベルのアプリケーション固有情報で定義されます。デフォルト値は、両方とも true です。

- 新しいコネクタ・プロパティ ConnectorID は、イベント分散をサポートするために使用できます。

リリース 1.1.x の新機能

このコネクタ・バージョン 1.1.x 用のリリース資料には、次の新規または訂正情報が記載されています。

- イベント通知サポート: 4 ページの『イベント処理コンポーネント』、8 ページの『アプリケーション・イベントの処理』、19 ページの『アプリケーション・イベントを処理するために必要なインストール操作』、23 ページの『アプリケーション・イベントの処理コード』、25 ページの『イベントおよびアーカイブ表』。
- 4 つの新規コネクタ固有の構成プロパティ: 126 ページの『ConvertToPrimitiveFloat』、126 ページの『EventKeyDelimiter』、127 ページの『SetLangCode』、および 127 ページの『ReconnectSessionOnGetFail』。

第 1 章 概要

この章では、役立つアダプター関連用語の定義を示し、IBM WebSphere Business Integration Adapter for PeopleSoft のコネクター・コンポーネントについて説明します。コネクターにより、統合ブローカーが PeopleTools バージョン 8.14 から 8.43 を使用する PeopleSoft バージョン 8.x アプリケーションとビジネス・オブジェクトを交換できます。

コネクターは、アプリケーション固有のコンポーネントとコネクター・フレームワークから成り立っています。アプリケーション固有のコンポーネントには、特定のアプリケーションに応じて調整されたコードが含まれています。コネクター・フレームワークのコードは、すべてのコネクターに共通です。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントの間を中継します。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントの間で以下のサービスを提供します。

- ビジネス・オブジェクトの受信と送信
- 始動メッセージおよび管理メッセージの交換の管理

この章は、以下のセクションから構成されています。

- 『用語』
- 3 ページの『コネクター・コンポーネント』
- 4 ページの『コネクターの動作方法』

統合ブローカーとコネクターの関係の詳細については、「*IBM WebSphere InterChange Server システム管理ガイド*」または「*MQ Integrator Broker 用インプリメンテーション・ガイド*」を参照してください。

用語

本書では、以下の用語を使用します。

アダプター

WebSphere Business Integration システム内のコンポーネントであり、統合ブローカーとアプリケーションまたはテクノロジーとの間の通信をサポートするためにコンポーネントを提供します。アダプターには、常にコネクター、メッセージ・ファイル、および構成ツールが組み込まれています。Object Discovery Agent (ODA) またはデータ・ハンドラーを組み込むこともできます。

アダプター・フレームワーク

アダプターの構成と実行を行うために IBM が提供するソフトウェア。アダプター・フレームワークのランタイム・コンポーネントには、Java ランタイム環境、コネクター・フレームワーク、および Object Discovery Agent (ODA) ランタイムが組み込まれています。このコネクター・フレームワークには、新しいコネクターの開発に必要なコネクター・ライブラリー (C++ および Java) が組み込まれています。ODA ランタイムには、新しい ODA

の開発に必要なライブラリーが Object Development Kit (ODK) 内に組み込まれています。構成コンポーネントには、次のツールが含まれます。

- Business Object Designer、
- Connector Configurator、
- Log Viewer、
- System Manager、
- Adapter Monitor、
- Test Connector
- およびアダプターに関連付けられた Object Discovery Agent (ODA) (オプション)。

Adapter Development Kit (ADK)

アダプター開発用のいくつかのサンプルを提供する開発キットであり、サンプル・コネクタおよび Object Discovery Agent(ODA) が組み込まれています。

コネクタ

アダプターのコンポーネントであり、ビジネス・オブジェクトを使用して、イベントに関する情報を統合ブローカーに送信したり (イベント通知)、要求に関する情報を統合ブローカーから受信します (要求処理)。コネクタは、コネクタ・フレームワークおよびコネクタのアプリケーション固有コンポーネントで構成されます。

コネクタ・フレームワーク

コネクタのコンポーネントであり、コネクタのアプリケーション固有コンポーネントと統合ブローカーとの間の対話を管理します。このコンポーネントは、必要なすべての管理サービスを提供し、コネクタが必要とするメタデータをリポジトリから取得します。コネクタ・フレームワークのコードはすべてのコネクタで共通であり、Java で記述されていますが、C++ で記述されたアプリケーション固有コンポーネントをサポートするために C++ 拡張機能が組み込まれています。

コネクタ・コントローラー

コネクタ・フレームワークのサブコンポーネントであり、コラボレーションと対話します。コネクタ・コントローラーは InterChange Server 内で実行され、アプリケーション固有オブジェクトと汎用ビジネス・オブジェクトとの間のマッピングを開始し、ビジネス・オブジェクト定義に対するコラボレーションのサブスクリプションを管理します。

統合ブローカー

WebSphere Business Integration システムのコンポーネントであり、異種のアプリケーション間でデータを統合します。一般に、統合ブローカーは、データを送付する機能、統合プロセスを管理する規則のリポジトリ、各種アプリケーションへの接続性、および統合を容易にする管理機能などのさまざまなサービスを提供します。統合ブローカーの例としては、WebSphere Business Integration Message Broker および WebSphere Business InterChange Server があります。

WebSphere Business Integration システム

異種のソース間で情報を移動させてビジネス取引を実行したり、エンタープ

ライズ環境において異なるアプリケーション間で情報を処理および送付するエンタープライズ・ソリューション。 Business Integration システムは、1つの統合ブローカーと1つ以上のアダプターで構成されます。

コネクタ・コンポーネント

IBM WebSphere Business Integration Adapter for PeopleSoft には、コネクタと、少なくとも1つの PeopleSoft コンポーネントとコンポーネント・インターフェースが組み込まれています。図1に、コネクタと、PeopleSoft アプリケーションとの関連を示します。

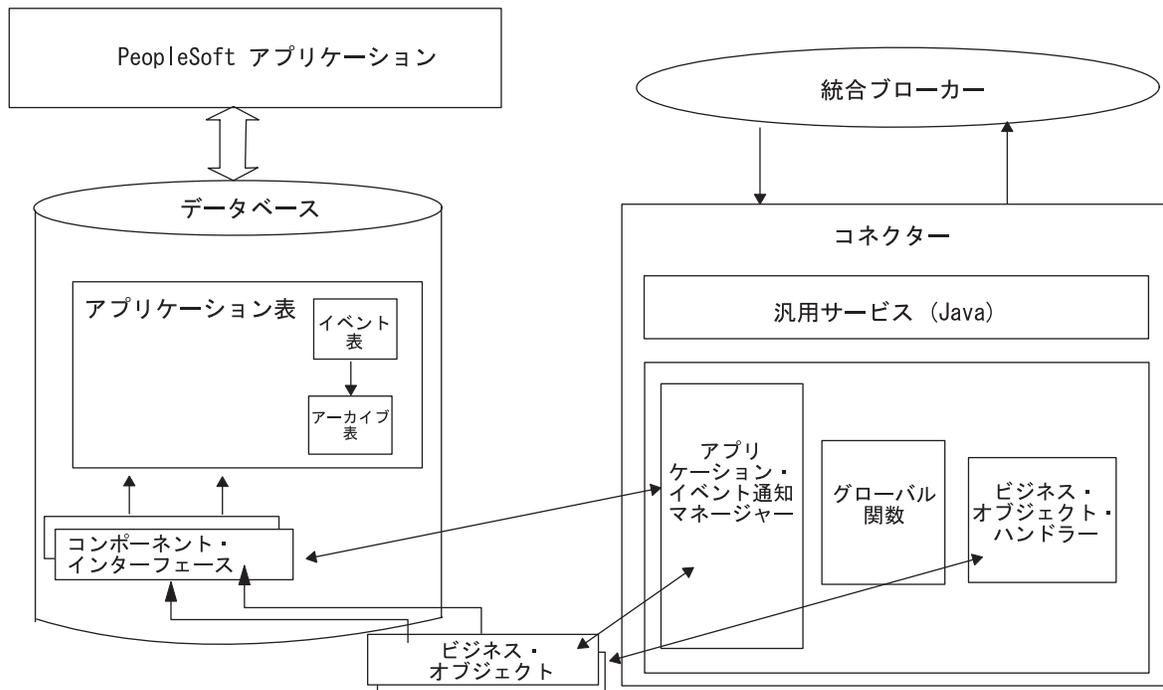


図1. コネクタ・アーキテクチャー

Connector for PeopleSoft 8

Java で作成されたコネクタは、アダプター用 IBM ビジネス・インテグレーション・システム規格に準拠します。コネクタは PeopleSoft セッション・オブジェクトを確立し、標準コネクタ・メソッドを使用してコンポーネントを処理します。

ビジネス・コンポーネント、コンポーネント・インターフェース、およびレコード

コネクタは、処理する階層構造の各ビジネス・オブジェクトについて PeopleSoft ビジネス・コンポーネントとコンポーネント・インターフェースを必要とします。

アダプターには、PeopleSoft 固有のビジネス・オブジェクトや、各ビジネス・オブジェクトに関連させる必要のあるコンポーネントやコンポーネント・インターフェースは組み込まれていません。これらのオブジェクトは、コネクタを実装するユ

ユーザーが作成する必要があります。コンポーネント・インターフェースとそれに対応するビジネス・オブジェクトの作成の詳細については、47 ページの『コンポーネント・インターフェースとビジネス・オブジェクトの関係』を参照してください。必要なクラスとメソッドの作成の詳細については、52 ページの『API の生成』を参照してください。

注: この資料では、ディレクトリー・パスに円記号 (¥) を使用します。UNIX システムの場合には、円記号 (¥) はスラッシュ (/) に置き換えてください。すべてのファイルのパス名は、ご使用のシステムで製品がインストールされているディレクトリーを基準とした相対パス名です。

イベント処理コンポーネント

イベント通知を使用可能にするために、アダプターに CW_EVENT_vX プロジェクトが用意され、これには次のものが組み込まれています。

- フィールド

フィールドにイベント情報が格納されます。

- レコード

レコードには、イベント表 (CW_EVENT_TBL)、アーカイブ表 (CW_ARCHIVE_TBL)、および関数ライブラリー (FUNCLIB_CW) が用意されています。関数ライブラリーには、SavePostChg() PeopleCode イベントでコンポーネントおよびレコードから呼び出されるコネクタ固有のイベント通知関数が格納されています。詳細については、8 ページの『アプリケーション・イベントの処理』、23 ページの『cw_publish_event() および cw_publish_future_dated_events() 関数』、および 25 ページの『イベントおよびアーカイブ表』を参照してください。

- コンポーネント

CW_EVENT_BC コンポーネントにより、イベントをオンラインで参照できます。また、イベント処理に必要なコンポーネント・インターフェースの適切な構造体も用意しています。コンポーネント・インターフェースにより、コネクターの処理に必要な CW_EVENT_TBL レコードのフィールドが公開されます。CW_EVENT_BC コンポーネントには、CW_EVENT_TBL レコードがあります。

- コンポーネント・インターフェース

CW_EVENT_CI コンポーネント・インターフェースにより、イベントの処理に必要なプロパティとメソッドとしてコンポーネントのフィールドとレコードが公開されます。

コネクターの動作方法

このセクションでは、以下について説明します。

- 5 ページの『PeopleSoft アプリケーションとの対話』
- 5 ページの『ビジネス・オブジェクト要求の処理』
- 8 ページの『アプリケーション・イベントの処理』

PeopleSoft アプリケーションとの対話

始動時に、PeopleSoft アプリケーション・サーバーに接続するために使用するセッション・オブジェクトを作成します。アプリケーション・サーバーに接続すると、サポート・ビジネス・オブジェクトに対応するすべてのコンポーネント・インターフェースの API に、コネクターがアクセスできます。サーバーには、イベント通知のためにアダプターに組み込まれている、PeopleCode およびアプリケーション・デザイナー・オブジェクトへのアクセス機能も用意されています。

各コンポーネント・インターフェース (および、それに関連するビジネス・コンポーネント、レコード、フィールド、スクロール、および PeopleCode) には、コネクターが PeopleSoft に関する階層構造の WebSphere ビジネス・オブジェクトを処理するために必要な情報がすべて格納されています。各コンポーネント・インターフェースはビジネス・コンポーネントのデータと処理ロジックをカプセル化しているため、コネクターはこの処理ロジックを複製しません。例えば、コネクターは重複レコードの検査、編集表の検証、またはセキュリティーを明示的に処理する必要はありません。

コネクター内、または PeopleSoft アプリケーション内でのオンライン処理時にエラーが発生した場合、コネクターのアプリケーション固有のコンポーネントからコネクター・フレームワークに戻りコード FAIL が送られ、さらにそれがコネクター・フレームワークから統合ブローカーに送られます。コネクターとアプリケーション・サーバーとの接続が失われると、コネクターのアプリケーション固有のコンポーネントにより、戻りコード APPRESPONSETIMEOUT が送られて terminate() メソッドが呼び出されます。

ビジネス・オブジェクト内のデータをコネクターが処理する方法の詳細については、45 ページの『第 4 章 コネクター用のビジネス・オブジェクトについて』を参照してください。

ビジネス・オブジェクト要求の処理

コネクターは、アプリケーション内のデータを変更するビジネス・オブジェクト要求を受け取ると、階層ビジネス・オブジェクトを再帰的に処理します。つまり、コネクターは、ビジネス・オブジェクトに関連するコンポーネント・インターフェースのすべてのレベルのデータを処理するまで、子ビジネス・オブジェクトをそれぞれ処理します。

コネクターは、統合ブローカーからのビジネス・オブジェクト要求を処理する場合、次の順序でセッション・オブジェクトから PeopleSoft API を呼び出します。

1. コネクターは、getComponent("ciName") メソッドを使用して、ビジネス・オブジェクトに関連したコンポーネント・インターフェースを戻します。コンポーネント・インターフェースの名前は、各ビジネス・オブジェクトのビジネス・オブジェクト・レベルでアプリケーション固有情報のプロパティに格納されます。
2. ビジネス・オブジェクトでデータの作成または変更が要求されると、コネクターはアプリケーションに値の挿入または変更を行います。ビジネス・オブジェクトによりデータの検索が要求されると、コネクターはアプリケーションから値を取得します。

- 作成または更新要求の場合、コネクタは `setFieldName(value)` メソッドを使用して、ステップ 1 で値を取得したコンポーネント・インターフェースの各フィールドを設定します。

キー属性のアプリケーション固有情報により PeopleSoft アプリケーションが固有 ID を生成することが指定されると、コネクタは、属性の値としてストリング NEXT を指定して、ビジネス・オブジェクトをアプリケーションに送ります。PeopleSoft 用の WebSphere ビジネス・オブジェクトでの、ストリング NEXT の使用の詳細については、69 ページの『属性レベルのアプリケーション固有情報』を参照してください。

ビジネス・オブジェクトのアプリケーション固有情報で `setInteractiveMode` を `true` に定義する場合、各フィールドの入力または終了で、基本となるコンポーネントの関連したビジネス・ロジックが起動され、これによりエラーが PeopleSoft PSMMessage コレクション・キューにただちにパブリッシュされます。

- 検索要求の場合、コネクタは `getFieldName()` メソッドを、値を指定しないで使用します。

ビジネス・オブジェクト内のデータをコネクタが処理する方法の詳細については、45 ページの『第 4 章 コネクタ用のビジネス・オブジェクトについて』を参照してください。コネクタ固有のプロパティの詳細については、125 ページの『付録 B. コネクタ固有のプロパティ』を参照してください。

3. すべての子ビジネス・オブジェクトを含め、ビジネス・オブジェクト全体に関する作成または更新要求を処理した後、コネクタはコンポーネント・インターフェースの `Save()` メソッドを呼び出します。成功すると、`Save()` メソッドにより 1 つの COMMIT ステートメントが実行されます。ビジネス・オブジェクトのアプリケーション固有情報で `setInteractiveMode` が `false` と定義される場合、`Save()` メソッドを呼び出すと、基本となるコンポーネントのレコードとフィールドに関連したすべての `FieldEdit` ビジネス・ロジックが起動されます。すべての PeopleCode エラーが、PSMessage コレクション・キューにパブリッシュされます。
4. 作成要求時に PeopleSoft システムで固有 ID が生成される場合、コネクタは標準の PeopleSoft 自動番号付け機能 (組み込み機能) を使用して、一番最近に使用された ID を検索し、ビジネス・オブジェクトに新しい ID を設定します。
5. コネクタは、関連するコンポーネント・インターフェースに用意されているすべてのビジネス・ロジックを使用して処理します。

コネクタがビジネス・オブジェクト要求を処理する際に接続エラーが検出されると、コネクタのアプリケーション固有コンポーネントにより致命的エラーがログに記録され、戻りコード `APPRESPONSETIMEOUT` が送信されて、電子メール通知が起動されます。その後、コネクタは終了します。

次のセクションでは、要求動詞の処理について説明します。

- 7 ページの『作成要求の処理』
- 7 ページの『検索要求の処理』
- 7 ページの『更新要求の処理』
- 7 ページの『削除要求の処理』

作成要求の処理

統合ブローカーが Create 動詞を使用してビジネス・オブジェクト要求を送信すると、コネクタは PeopleSoft の Create() メソッドを使用してコンポーネント・インターフェースの新しいインスタンスを作成します。トランザクションの進行で、インスタンスからすべての PeopleSoft ビジネス・ロジックが渡される場合、インスタンスはアプリケーションに保管されます。アプリケーションに作成されるオブジェクトに、すべての子ビジネス・オブジェクトなどの、ビジネス・オブジェクトに格納されるすべての値が格納されます。

作成操作の処理の詳細については、57 ページの『作成操作』を参照してください。

検索要求の処理

統合ブローカーが Retrieve 動詞を使用してビジネス・オブジェクト要求を送信すると、コネクタは PeopleSoft の Get() メソッドを使用して対応するコンポーネント・インターフェースの固有のインスタンスが存在することを確認します。この Get() メソッドによりコンポーネント・インターフェースがインスタンス化され、コネクタがその値をビジネス・オブジェクトにロードできるようになります。コネクタが統合ブローカーに戻すビジネス・オブジェクトは、コンポーネント・インターフェースのインスタンスと正確に一致します。

つまり、統合ブローカーに戻されるビジネス・オブジェクトの各単純属性の値は、コンポーネント・インターフェースの対応するプロパティ・フィールドの値と一致します。また、戻されたビジネス・オブジェクトが階層構造の場合、各配列内の個々のビジネス・オブジェクト数は、その配列に関するコンポーネント・インターフェース内のレベルまたはコレクションの数に一致します。

検索操作の処理の詳細については、59 ページの『検索操作』を参照してください。

更新要求の処理

統合ブローカーが Update 動詞を使用してビジネス・オブジェクト要求を送信すると、コネクタはコンポーネント・インターフェースの既存のインスタンスを変更します。トランザクションの進行で、インスタンスからすべての PeopleSoft ビジネス・ロジックが渡される場合、インスタンスはアプリケーションに保管されます。

アプリケーションで更新されるオブジェクトは、要求ビジネス・オブジェクトと正確に一致します。コネクタによりすべての単純なプロパティ・フィールドが更新されます。ただし、要求ビジネス・オブジェクト内の対応する属性に値 CxIgnore がある場合は除きます。要求ビジネス・オブジェクト内に格納されているすべての子ビジネス・オブジェクトを挿入します。コネクタは、その KeepRelationship アプリケーション固有情報パラメーターの値に基づいて、要求ビジネス・オブジェクトに存在しない子ビジネス・オブジェクトを削除または保持します。

更新操作の処理の詳細については、62 ページの『更新操作』を参照してください。

削除要求の処理

PeopleSoft がトランザクションの削除をサポートしないため、コネクタもサポートしません。論理削除の処理の標準の振る舞いは、Update 動詞を使用してビジネス・オブジェクトの EffectiveStatus 属性の状況を「I」(非アクティブ)に変更することです。

ただし、コネクタがオブジェクト全体の削除をサポートするようにするには、次を実行します。

1. PeopleSoft 内の削除を実行するユーザー定義の PeopleCode メソッドを作成する。
2. 適切なコンポーネント・インターフェースでこのメソッドを公開する。

アプリケーション・イベントの処理

イベント通知には、次の 3 つの主要なプロセスが関係します。

- 『イベントのパブリッシュ』：イベントはコネクタのイベント表 (CW_EVENT_TBL) にパブリッシュされます。この表は PeopleSoft データベースに格納されます。
- 『イベントのポーリング』：コネクタは PeopleSoft API を使用してイベント表をポーリングします。
- 9 ページの『イベント・アーカイブ』：コネクタはイベントをアーカイブ表 (CW_ARCHIVE_TBL) にアーカイブします。

イベントのパブリッシュ

イベントをコネクタのイベント表にパブリッシュするために、PeopleSoft アプリケーションはアダプターに組み込まれている PeopleCode オブジェクトおよびアプリケーション・デザイナー・オブジェクトを使用します。これらのオブジェクトの詳細については、4 ページの『イベント処理コンポーネント』を参照してください。

イベントのパブリッシュ・プロセスでは、`cw_publish_event()` 関数または `cw_publish_future_dated_event()` 関数を使用します。これらの関数は、FUNCLIB_CW レコードの FieldFormula イベントに格納されています。これらの関数のいずれかを宣言し、このイベントに関与しているコンポーネントの `SavePostChg()` PeopleCode からその関数を呼び出す必要があります。

注： この手順では、コンポーネント・インターフェースを使用しません。

詳細については、23 ページの『`cw_publish_event()` および `cw_publish_future_dated_events()` 関数』を参照してください。

イベントのポーリング

コネクタは、構成可能で定期的な間隔でイベント表をポーリングします。イベントの検索は、最初は状況により、次にコネクタ・プロパティ `ConnectorID` の値により行います。この値が `null` またはブランクの場合、プロパティは検索に使用されません。コネクタがイベントを処理すると、状況をただちに `INPROGRESS` (値 3) に更新します。コネクタの初期設定時は、保留中の `INPROGRESS` イベントは `READYFORPOLL` (値 0) にリセットされます。

コネクタは `CW_EVENT_CI` コンポーネント・インターフェースの `Find()` メソッドを使用してイベント表をポーリングします。このメソッドは、状況が `READYFORPOLL` であるイベントのコレクションを戻します。コネクタはコレクションをループ処理して、`CW_EVENT_TBL` にリストされているすべてのビジネス・オブジェクトの名前を取得します。

コネクタは API を使用してプロパティ値を設定および取得して、コレクションで戻された各イベントのイベント情報を収集します。また、サブスクライブされるビジネス・オブジェクトを検査して判別します。使用する統合ブローカーに固有のサブスクリプション情報については、ブローカーのインプリメンテーション・ガイドを参照してください。

- サブスクライブされる各ビジネス・オブジェクトに関して、コネクタはそのオブジェクトが関連したコンポーネント・インターフェース名を検索します。コネクタは `CW_OBJ_KEYS` フィールドにリストされているキー値を使用して、イベントに関係するコンポーネント・インターフェースをインスタンス化し、情報をアプリケーション固有のビジネス・オブジェクトにコピーします。ビジネス・オブジェクトを作成後、コネクタはそれを統合ブローカーに渡して、イベントをアーカイブします。
- アンサブスクライブされるビジネス・オブジェクトとエラーを生成するビジネス・オブジェクトの場合、コネクタはイベントをアーカイブ表に移動して、その状況を更新します。
- このコネクタは、アウトバウンド・オブジェクトの `Delete` 動詞をサポートします。ビジネス・オブジェクト・インスタンスのキー値を設定し、そのオブジェクトを WBI 統合ブローカーに送信します。

詳細については、25 ページの『イベントおよびアーカイブ表』を参照してください。

イベント・アーカイブ

`CW_EVENT_CI` コンポーネント・インターフェースがインスタンス化される間に、コネクタは処理する各イベントに関してユーザー定義のメソッド

`cw_archive_events()` を呼び出します。PeopleCode 形式のこのメソッドにより、アーカイブ・フラグが `Y` に設定されます。このフラグが設定されると、ユーザーが `Save()` メソッドを呼び出す際に、`SavePostChg()` PeopleCode によりイベントがアーカイブされます。アーカイブ表にイベントが処理された日付と時刻も記録されます。

注: `SavePostChg()` PeopleCode は、プロジェクトに組み込まれています。

アーカイブ表は、イベント・ヒストリーの照会またはイベント処理の問題のトラブルシューティングに使用することができます。例えば、アンサブスクライブされるイベントはすべて、`unsubscribed` の状況になっています。

イベントの処理に関連した潜在的な障害ポイントが存在するため、イベント管理プロセスでは、イベントがアーカイブ表に挿入されるまで、イベント表からイベントを削除しません。

イベント通知プロセス・フロー

以下のステップでは、イベント通知プロセスを説明します。

1. ユーザーがコンポーネントに変更をオンラインで保管すると、`SavePostChg()` メソッドによって `cw_publish_event()` 関数または `cw_publish_future_dated_events()` 関数が呼び出されます。
2. 4 つのパラメーターを使用する `cw_publish_event()` 関数で、次が実行されません。

- 最初のパラメーターからアプリケーション固有のビジネス・オブジェクト名を取得します。
- アプリケーション固有のビジネス・オブジェクトに対応するコンポーネント・インターフェースをインスタンス化するために必要な各キーの名前を取得します。2 番目のパラメーターからキー名とその値を取得します。
- CW_CONN_ID フィールドに 4 番目のパラメーターの値を設定します。このパラメーターは ConnectorID コネクタ・プロパティにも対応します。デフォルト値は PeopleSoftConnector です。
- イベントの固有 ID を生成します。
- PeopleSoft システム変数 %Mode を評価して、イベントに関連した動詞を識別します。
- イベントを CW_EVENT_TBL に挿入します。

注: cw_publish_future_dated_events() 関数は、cw_publish_event()関数と同様の処理を実行しますが、追加のパラメーターを使用する点が異なります。保管するコンポーネントの任意の日付属性 (通常は有効日付) を渡して、将来の日付が到来した時点でのみ、その日付のイベントをポーリングするようにするには、この関数を使用します。有効日付が将来の日付である場合、イベント状況は 99 に設定され、イベント日付は将来の日付に設定されます。そうでない場合、状況は 0 に設定され、日付はシステム日付に設定されます。

詳細については、23 ページの『cw_publish_event() および cw_publish_future_dated_events() 関数』を参照してください。

3. コネクタは、その PollFrequency 構成プロパティで指定された間隔でイベント表からイベントをポーリングして、イベントを選別します。しかしその数は PollQuantity 構成プロパティで指定された数を超えません。ポーリングを行うために、コネクタは CW_EVENT_CI コンポーネントをインスタンス化して、Find() メソッドを呼び出します。

注: コネクタは、要求処理に使用されたスレッドとは別のスレッドを使用します。

4. イベント表から取り出された各レコードについて、コネクタはレコードがイベント・データをサブスクライブするかどうかを検査します。

データがサブスクライブされない場合、コネクタはイベント・レコードを CW_EVENT_TBL から CW_ARCHIVE_TBL に移動し、状況を unsubscribed にします。

5. データがサブスクライブされる場合、コネクタはイベント・レコードのキー・フィールドを使用して該当するコンポーネント・インターフェースをインスタンス化し、ここからイベントの完全なデータを検索します。
6. コネクタは、コンポーネント・インターフェースから完全なイベント・データを検索した後で、該当するアプリケーション固有のビジネス・オブジェクトをインスタンス化します。
7. コネクタは、アプリケーション固有のビジネス・オブジェクトを統合ブローカーに送信します。

8. コネクタは `cw_archive_events()` 関数を呼び出します。この関数でアーカイブ・フラグを `Y` に設定し、状況と処理時間を適切に設定します。アーカイブ・フラグが設定されると、`SavePostChg()` `PeopleCode` によりイベント情報がアーカイブ表にコピーされ、イベント表からイベントが削除されます。

ロケール依存型データの処理

コネクタは国際化されているので、2 バイト文字セットに対応でき、指定された言語でメッセージ・テキストをデリバリーできます。コネクタがある文字コード・セットを使用する場所から別の文字コード・セットを使用する場所にデータを転送する場合、文字変換を行ってデータの意味を保持します。

Java 仮想マシン (JVM) 内の Java ランタイム環境では、データを Unicode 文字コード・セットで表現します。Unicode は周知の文字コード・セット (単一バイトとマルチバイトの両方) で文字をエンコードします。WebSphere Business Integration システムのほとんどのコンポーネントは Java で作成されています。したがって、ほとんどの統合コンポーネント間でデータが転送される場合、文字変換は必要ありません。

該当する国または地域の適切な言語でエラーおよび情報メッセージをログに記録するには、使用する環境の `Locale` 標準構成プロパティを設定します。構成プロパティの詳細については、105 ページの『付録 A. コネクタの標準構成プロパティ』を参照してください。

第 2 章 コネクタのインストール

この章では、IBM WebSphere Business Integration Adapter for PeopleSoft のコネクタ・コンポーネントのインストールおよび構成の方法と、PeopleSoft アプリケーションをコネクタが動作するように構成する方法について説明します。

この章は、以下のセクションから構成されています。

- 『アダプター環境』
- 14 ページの『前提条件』
- 16 ページの『アダプターと関連ファイルのインストール』
- 17 ページの『インストールの検査』
- 18 ページの『コネクタ用のアプリケーションを使用可能にする』

アダプター環境

アダプターをインストール、構成、使用する前に、環境要件を理解しておく必要があります。

ブローカーの互換性

アダプターが使用するアダプター・フレームワークは、アダプターと通信する統合ブローカーのバージョンとの互換性を備えている必要があります。Adapter for PeopleSoft バージョン 3.0 は、以下のアダプター・フレームワークと統合ブローカーでサポートされています。

- **アダプター・フレームワーク:** WebSphere Business Integration Adapter Framework、バージョン 2.1、2.2、2.3.x、および 2.4
- **統合ブローカー:**
 - WebSphere InterChange Server、バージョン 4.11、4.2、4.21、および 4.22
 - WebSphere MQ Integrator、バージョン 2.1.0
 - WebSphere MQ Integrator Broker、バージョン 2.1.0
 - WebSphere Business Integration Message Broker、バージョン 5.0
 - WebSphere Application Server Enterprise、バージョン 5.0.2 (WebSphere Studio Application Developer Integration Edition、バージョン 5.0.1 と併用)

例外については、「リリース情報」を参照してください。

注:

統合ブローカーのインストール手順およびその前提条件については、次の資料を参照してください。

WebSphere InterChange Server (ICS) については、「システム・インストール・ガイド (UNIX 版)」または「システム・インストール・ガイド (Windows 版)」を参照してください。

Message Brokers (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) の場合は、「WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド」およびそれぞれの Message Broker のインストールに関する資料を参照してください。一部の資料は次の Web サイトにあります。

<http://www.ibm.com/software/integration/mqfamily/library/manualsa/>

WebSphere Application Server については、「アダプター実装ガイド (WebSphere Application Server)」および次の資料を参照してください。

<http://www.ibm.com/software/webservers/appserv/library.html>

アダプター・プラットフォーム

アダプターは、次のソフトウェア・プラットフォームでサポートされます。

- Windows 2000
- AIX 5.1、5.2
- Solaris 8
- HP UX11i

前提条件

IBM WebSphere Business Integration Adapter for PeopleSoft を使用する前に、使用するシステムに必要なソフトウェアがインストールされていることを確認します。また、コネクターのユーザー・アカウントの作成、環境変数のマッピング、および PeopleSoft API の生成などの操作を実行する必要もあります。このセクションでは、以下について説明します。

- 『必要なソフトウェア』
- 15 ページの『コネクターをインストールするために必要な操作』

必要なソフトウェア

指定されたコンポーネント・インターフェースとその基になっている PeopleSoft ビジネス・ロジックにアクセスするために、コネクターには 2 層の PeopleSoft API クラスが必要です。コネクターを使用する前に、次のものをインストールします。

- `psjoa.jar`: コネクターはこの API を使用して、BEA システムの Jolt ポートを經由してアプリケーション・サーバーに接続します。コネクターはまた、この API を使用してアプリケーション・サーバー経由でオブジェクトを送信します。このファイルは、PeopleTools の現行バージョンに組み込まれており、次のディレクトリーに格納されています。

UNIX

`$PS_HOME/web/PSJOA`

UNIX の終り

Windows

```
%PS_HOME%\web\PSJOA
```

Windows の終り

このファイルをコネクターのインストール・ディレクトリーに保管します。PeopleSoft の API ファイルの検索の詳細については、PeopleSoft の「インストール・ガイドおよび管理ガイド」を参照してください。

- コンポーネント・インターフェース API: セッション・オブジェクトが作成され、psjao.jar を使用して PeopleSoft アプリケーション・サーバーへの接続が確立された後でのみ、コネクターはこの層にアクセスします。コンポーネント・インターフェース API には、コンポーネント・インターフェースの公開されたオブジェクトと PeopleCode メソッドのすべてに対するアクセス機能が用意されています。アプリケーション・デザイナーでこの API を手動で生成する必要があります (52 ページの『API の生成』を参照)。

重要: 使用する統合ブローカーに固有のインストール前提条件については、以下を参照してください。コネクターのコンポーネントのインストール方法の詳細については、「*WebSphere MQ Integrator Broker* 用インプリメンテーション・ガイド」、「*IBM WebSphere InterChange Server* システム・インストール・ガイド (Windows 版)」または「*IBM WebSphere InterChange Server* システム・インストール・ガイド (UNIX 版)」を参照してください。

コネクターをインストールするために必要な操作

このセクションでは、コネクターを使用する前に実行する必要がある、必須操作について説明します。

- PeopleSoft にコネクターのユーザー・アカウントを作成します。詳細については、16 ページの『PeopleSoft のユーザー・アカウントの作成』を参照してください。
- PeopleSoft ユーザー・アカウントがタイムアウトにならないように構成します。詳細については、16 ページの『ユーザー・アカウントのタイムアウトの構成』を参照してください。
- ICS が統合ブローカーの場合のみ: 以前にインストールされたコネクターのすべてのバージョンを削除します。また、PeopleSoft でコネクターに対して行われたカスタマイズもすべて削除します。例えば、コネクターに関連するメニュー、ページ、レコード、メッセージ定義コンポーネント、およびコンポーネント・インターフェースをすべて削除します。コネクターのイベント表およびアーカイブ表も削除します。通常、これらのオブジェクトはアプリケーション・デザイナーでオープンできるコネクター固有のプロジェクトに格納されています。

注: WebSphere Business Integration Adapter を、既存の ICS コネクターのインストール・システムと同じマシンにインストールすると、IBM WebSphere システム環境変数が指定の製品パスを指すように手動で変更しないかぎり、ICS コネクターは稼働しません。

- コンポーネント・インターフェース API を生成します。詳細については、52 ページの『API の生成』を参照してください。

PeopleSoft のユーザー・アカウントの作成

コネクタをインストールする前に、PeopleSoft アプリケーションのユーザー・アカウントを作成する必要があります。アカウントは次のとおりです。

- 有効な PeopleSoft ユーザー名とパスワードであれば何でも可能です。
- PeopleSoft システム内の該当するページ、コンポーネント、およびコンポーネント・インターフェースからデータを検索、挿入、更新、および削除する特権が必要です。例えば、顧客データを処理するコネクタの場合、コネクタのユーザー・アカウントには関連する顧客コンポーネントを使用する特権が必要です。
- 一般的な PeopleSoft Security を使用します。詳細については、PeopleBooks の「Administrator Tools」の『Security』セクションを参照してください。
- セッションがタイムアウトにならないように構成する必要があります。詳細については、『ユーザー・アカウントのタイムアウトの構成』を参照してください。

ユーザー・アカウントのタイムアウトの構成

コネクタはコンポーネント・インターフェース (およびアプリケーション・サーバー上のコンポーネント・プロセッサ) を使用して、PeopleSoft データベース・サーバーとの間でビジネス・オブジェクトの挿入および抽出を行うため、アプリケーション・サーバーが実行している必要があります。ユーザー・アカウントがタイムアウトになると、コンポーネント・プロセッサが終了し、コネクタは PeopleSoft システムにアクセスを試行するたびに (つまり、ポーリング、作成、検索、および更新のたびに) エラー・メッセージをログに記録します。

ユーザー・アカウントがタイムアウトにならないように構成するには、次の手順で行います。

1. PeopleSoft アプリケーションで、「Go」メニュー・オプションの「PeopleTools」を選択する。
2. 「PeopleTools」メニュー・オプションの「Maintain Security」を選択する。
3. 「Use」メニュー・オプションの「Permission Lists」を選択する。
4. 「Permission Lists」メニュー・オプションの「Sign-On Times」を選択する。
5. アクセス権のリストから、コネクタのユーザー・アカウントに関連するアクセス権を選択する。
6. 「General」タブで、「Never Time-Out」を選択し、「Can Start Application Server」を選択していない場合は選択する。

アダプターと関連ファイルのインストール

WebSphere Business Integration Adapter 製品のインストールについては、「*WebSphere Business Integration Adapters* インストール・ガイド」を参照してください。この資料は、次の Web サイトの WebSphere Business Integration Adapters Infocenter にあります。

<http://www.ibm.com/websphere/integration/wbiadapters/infocenter>

インストールの検査

このセクションでは、UNIX システムまたは Windows システムに製品をインストールした後のファイル構造について説明します。

インストール済みファイルの検査 (UNIX システムの場合)

UNIX システムでインストールを検査するには、アダプターのインストール・ディレクトリーにあるファイルを、次に示すファイルと比較します。

表 1 に、コネクターが使用する UNIX ファイル構造の説明を示します。

表 1. コネクター用としてインストールされた UNIX ファイルのファイル構造

\$ProductDirS のサブディレクトリー	説明
connectors/PeopleSoft	<ul style="list-style-type: none">コネクターの CWPeopleSoft.jar と start_PeopleSoft.sh ファイルが格納されます。 <p>コネクターの始動スクリプトは、汎用コネクター・マネージャー・スクリプトから呼び出されます。製品インストーラーの「コネクター構成」画面から、「インストール」をクリックすると、インストーラーによりこのコネクター・マネージャー・スクリプトのカスタマイズされたラッパーが作成されます。</p> <p>コネクターが WebSphere Interchange Server のブローカーと連動する場合、このカスタマイズされたラッパーを使用してコネクターを始動および停止します。コネクターが WebSphere MQ Integrator のブローカーと連動する場合、このカスタマイズされたラッパーはコネクターを始動するためのみに使用します。コネクターを停止するには、mqsiremotestopadapter を使用します。</p> <ul style="list-style-type: none">EventNotificationInstall.exe という名前の実行可能ファイルが格納されます。このファイルにはイベント通知オブジェクトが組み込まれています。
/lib	このディレクトリーに WBIA.jar ファイルが含まれていることを確認してください。このファイルは、ADK 2.2.0 に対応します。以前のバージョンの ADK が必要な場合は、「コネクター開発ガイド」を参照してください。
connectors/PeopleSoft/dependencies	<ul style="list-style-type: none">現行バージョンの PeopleTools からコピーする必要がある psjoa.jar ファイルが格納されます。PSFTCI.jar ファイルが格納されます。このファイルには、API をビルドしてコンパイルした後で PeopleSoft により生成される、すべての PeopleSoft コンポーネント・インターフェース・クラス・ファイルが格納されます。
connectors/messages /bin/APP/DATA	PeopleSoftConnector.txt ファイルが格納されます。 PeopleSoft Connector テンプレートが格納されます。これは、Connector Configurator を使用してコネクターの構成ファイルを作成する際に使用します。

インストール済みファイルの検査 (Windows システムの場合)

Windows システムでインストールを検査するには、アダプターのインストール・ディレクトリーにあるファイルを、次に示すファイルと比較します。

次の表に、コネクターが使用する Windows ファイル構造の説明を示します。

表 2. コネクター用としてインストールされた Windows ファイルのファイル構造

<code>%ProductDirS%</code> のサブディレクトリー	説明
<code>¥connectors¥PeopleSoft</code>	<ul style="list-style-type: none">コネクターの <code>CWPeopleSoft.jar</code> と <code>start_PeopleSoft.bat</code> ファイルが格納されます。<code>EventNotificationInstall.exe</code> という名前の実行可能ファイルが格納されます。このファイルにはイベント通知オブジェクトが組み込まれています。
<code>..¥bin</code>	このフォルダーに <code>CWConnEnv.bat</code> ファイルが含まれていることを確認してください。
<code>..¥lib</code>	このディレクトリーに <code>WBIA.jar</code> ファイルが含まれていることを確認してください。このファイルは、ADK 2.2.0 に対応します。以前のバージョンの ADK が必要な場合は、「コネクター開発ガイド」を参照してください。
<code>¥connectors¥PeopleSoft¥dependencies</code>	<ul style="list-style-type: none">現行バージョンの <code>PeopleTools</code> からコピーする必要のある <code>psjoa.jar</code> ファイルが格納されます。<code>PSFTCI.jar</code> ファイルが格納されます。このファイルには、API をビルドしてコンパイルした後で <code>PeopleSoft</code> により生成される、すべての <code>PeopleSoft</code> コンポーネント・インターフェース・クラス・ファイルが格納されます。
<code>¥connectors¥messages</code>	<code>PeopleSoftConnector.txt</code> ファイルと、 <code>PeopleSoftConnector_II_IT.txt</code> ファイル (言語 (<code>II</code>) および国/地域 (<code>IT</code>) に固有のメッセージ・ファイル) が格納されます。
<code>¥bin¥APP¥DATA</code>	<code>PeopleSoft Connector</code> テンプレートが格納されます。これは、 <code>Connector Configurator</code> を使用してコネクターの構成ファイルを作成する際に使用します。

コネクター用のアプリケーションを使用可能にする

`PeopleSoft` アプリケーションでコネクターを使用するには、事前に次を実行する必要があります。

- ご使用の `PeopleSoft` アプリケーションに必要なフィールド定義、ページ、レコード、コンポーネント、およびコンポーネント・インターフェースを定義します。
- アダプターに組み込まれているプロジェクトをインポートします。このプロジェクトには、イベント処理、必要な表のビルド、および API ファイルのビルドに必要なコンポーネントが格納されています。

注: プロジェクトのインポートは、コネクターを使用してイベントを処理する場合にのみ必要です。

このセクションでは、以下について説明します。

- 19 ページの『アプリケーション・イベントを処理するために必要なインストール操作』
- 23 ページの『アプリケーション・イベントの処理コード』

アプリケーション・イベントを処理するために必要なインストール操作

このセクションでは、コネクタを使用してアプリケーション・イベントを処理する場合に実行する必要がある操作について説明します。コネクタを初めて使用する前に、このインストールを実行してください。

イベント処理コンポーネントのインストールには、次の操作が関係します。

- 『プロジェクトのインポート』
- 20 ページの『必要なオブジェクトのビルド』
- 21 ページの『API ファイルのビルド』

プロジェクトのインポート

1. ファイルを実行して、アプリケーション・デザイナー・プロジェクト・ファイルを指定されたフォルダーに unzip する。

`%ProductDirS%\connectors\PeopleSoft\dependencies` ディレクトリーから、`EventNotificationInstall.exe` ファイルを選択します。

注: このファイルは Windows でのみ使用可能です。

デフォルト値でこのファイルを実行すると、`dependencies` ディレクトリー内に `Projects` ディレクトリーが作成されます。このディレクトリーには 2 つのプロジェクトが格納されています。ご使用の PeopleTools のバージョンに応じて、これらの 2 つのプロジェクトのうちのいずれかを使用してください。

- PeopleTools バージョン 8.14 から 8.40 まで (ただし 8.40 を含まない) の場合は `CW_EVENT_V140` を使用
 - PeopleTools バージョン 8.40 から 8.43 までの場合は `CW_EVENT_V150` を使用
2. PeopleSoft のアプリケーション・デザイナーを開き、「File」メニューの「Copy Project From File...」または「Copy From File: Select Project...」を選択する。
 3. 表示されるメニューで、プロジェクトを unzip するインポート・ディレクトリーを選択する。図 2 に、PeopleTools のいくつかのバージョンで表示される画面を示します。

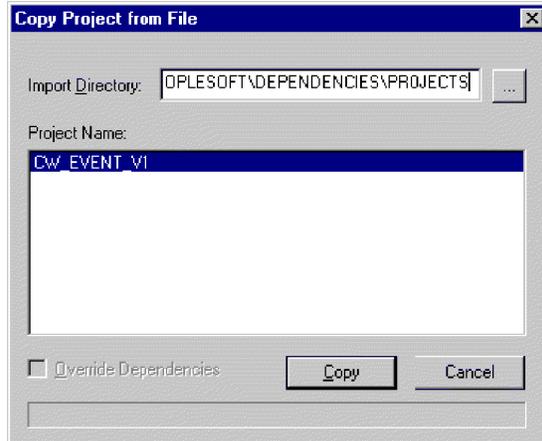


図2. プロジェクトのインポート

- 「コピー」をクリックして、4 ページの『イベント処理コンポーネント』にリストされているコンポーネントを持つコンポーネントのリストを確認する。

注: データベース・イベント DB2 を使用する場合は、CW_EVENT_NOT フィールドのデータ型を文字から数値に変更し、サイズを 15 に設定する必要があります。

必要なオブジェクトのビルド

イベント表、アーカイブ表、および関数ライブラリーをビルドするには、次の手順で行います。

- 「Build」メニューの「Project...」を選択して、表をビルドする。
- 表示されたダイアログ・ボックスで、CW_EVENT_TBL、FUNCLIB_CW、および CW_ARCHIVE_TBL が図3 のように表示されることを確認する。

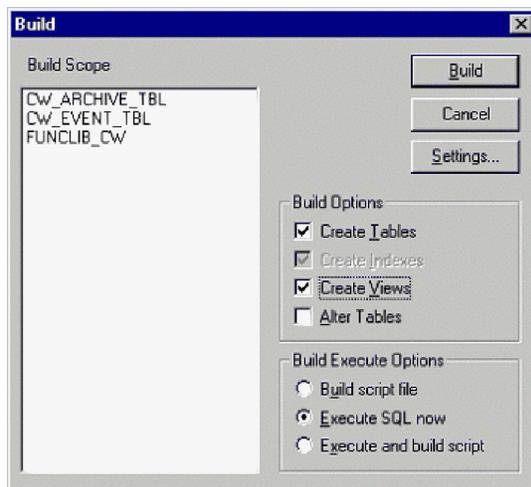


図3. 表のビルド

- ビルド・オプションの「Create Tables」と「Execute SQL now」を選択する。また、「Settings」ボタンをクリックし、表示されたウィンドウで、「create table if it already exists」と「recreate index if it already exists」が選択されていることを確認する。

4. 「Build」をクリックする。
5. 適切なデータベース特権を持つ ID を使用して、SQL エディターにログオンする。デフォルトの ID は SYSADM/SYSADM です。FUNCLIB_CW レコードの CW_EVENT_NOT フィールドを値ゼロで初期化します。このためには、次のステートメントを実行します。

```
INSERT INTO PS_FUNCLIB_CW (CW_EVENT_NOT) VALUES ('0');
```

API ファイルのビルド

CW_EVENT_CI API ファイル (コネクタがイベントのみを処理する場合に必要な) またはコンポーネント・インターフェース API ファイル (コネクタがイベントと要求の両方を処理する場合に必要な) をビルドするには、次の手順で行います。

1. プロジェクト・ウィンドウでコンポーネント・インターフェース CW_EVENT_CI を開き、「Build」メニューの「PeopleSoft API」を選択する。
2. Java クラス・パネルで、次を実行する。
 - 「Build」ボックスを選択する。
 - 「Directory containing PeopleSoft package:」フィールドにコネクタの dependencies ディレクトリへのパス (または、実装するすべての PeopleSoft API ファイルが存在するディレクトリ) を入力する。通常は、次のとおりです。

```
UNIX  
$ProductDirS/connectors/PeopleSoft/dependencies
```

```
UNIX の終り
```

```
Windows  
%ProductDirS%\connectors\PeopleSoft\dependencies
```

```
Windows の終り
```

3. 「Select APIs to Build:」フィールドから、CWEVENT_CI とそれに関連したコレクションを選択する。図 4 に、PeopleSoft 内の画面を示します。

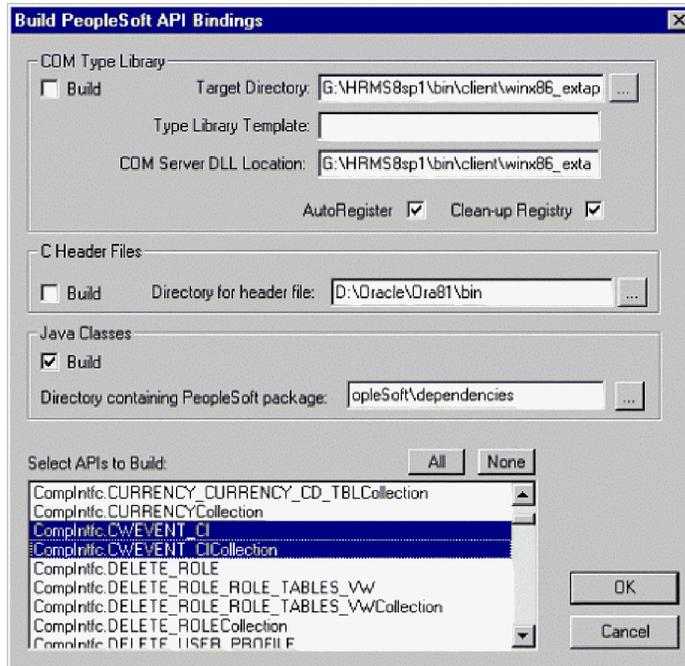


図4. API のビルド

4. 「OK」をクリックする。
5. 生成したすべての API ファイルをコンパイルして、PSFTCI.jar ファイルに追加する (存在する場合) か、または存在しない場合はこのファイルを作成する。

注: 始動スクリプトまたはバッチ・ファイルは、

%connectors%PeopleSoft%dependencies ディレクトリー内の PSFTCI.jar という名前のファイルからこれらの API ファイルを検索するように構成されています。これらの API クラスをコンパイル後、JAR ファイルを作成し、PSFTCI.jar という名前を付けます。この JAR ファイルを別のディレクトリー内に設定する場合は、始動スクリプトまたはバッチ・ファイルを変更して、これらの API クラスの正しい場所を指すようにします。

6. イベントに関連した各コンポーネントの SavePostChg() PeopleCode に、cw_publish_events() 関数宣言と関数呼び出しを設定する。詳細については、24 ページの『サンプルの PeopleCode 宣言および関数呼び出し』を参照してください。

注: 関数は、レコードではなくコンポーネントの SavePostChg() に設定されている必要があります。

7. 関数呼び出しで使用されるすべてのパラメーターを定義し、PeopleCode を挿入して、関数呼び出しの前にコンポーネントが変更されたかどうかを検査する。このコードにより、関数の不要な呼び出しがなくなります。また、%userid の検査を追加して、コネクターの往復ループ (つまり、コネクターが要求からのデータ変更を新規アプリケーション・イベントと解釈する永久ループの作成) を防止します。例については、24 ページの『サンプルの PeopleCode 宣言および関数呼び出し』を参照してください。

アプリケーション・イベントの処理コード

アダプターの FUNCLIB_CW 関数ライブラリーには、`cw_publish_event()` 関数および `cw_publish_future_dated_events()` 関数が組み込まれています。これらの関数の動作は似ており、コネクターのイベント表にイベントを挿入します。違いは、`cw_publish_future_dated_events()` は将来の有効日付を持つイベントのパブリッシュを使用可能にするのに対し、`cw_publish_event()` はこれを行わない点です。将来の有効日付を持つイベントを使用する場合は、`cw_publish_future_dated_events()` 関数を使用してください。

このセクションでは、これらの関数について説明し、これらの関数を呼び出すコード例を示します。

`cw_publish_event()` および `cw_publish_future_dated_events()` 関数

`cw_publish_event()` 関数は 4 つのパラメーターを取り、
`cw_publish_future_dated_events()` 関数は 5 つのパラメーターを取ります。

- **&BONAME:** イベント用に生成される PeopleSoft の WebSphere アプリケーション固有のビジネス・オブジェクト名が格納されます。(両方の関数で使用。)
- **&KEYLIST:** **&BONAME** パラメーターで指定されたビジネス・オブジェクトに対応するコンポーネント・インターフェースをインスタンス化するために必要なすべてのキー名が格納されます。この関数は `GetKey()` メソッドを使用してインスタンス化します。コネクターは名前と値の対を使用するので、キーの順序は関係ありません。しかし、名前はコンポーネント・インターフェースにリストされる名前と一致する必要があります。キーが複数の場合は、コロンまたは他の構成可能な区切り文字 (例えば、`SETID:DEPTID`) で分離します。詳細については、126 ページの『EventKeyDelimiter』を参照してください。(両方の関数で使用。)
- **&CWPRORITY1:** イベントの優先順位を定義します。デフォルト値は 2 です。このパラメーターにより、指定された優先順位が低いイベントより先に特定のイベントを処理することができます。(両方の関数で使用。)
- **&CONNID:** イベントを検索するコネクター・インスタンスの名前が格納されます。このパラメーターはイベント分散のために使用され、コネクターの `ConnectorID` プロパティに定義されます。イベント分散を使用しない場合は、値「`PeopleSoftConnector`」を使用します。(両方の関数で使用。)
- **&EFFDATE:** イベントに関連した日付。この値がシステム日付より大きい場合、`cw_publish_future_dated_events` 関数の `peoplecode` によってイベント日付が **&EFFDATE** 値に設定されます。(このパラメーターは `cw_publish_future_dated_events` で使用します。)

重要: これらのパラメーターは、正しい形式で、関数呼び出し時あるいはその前に定義する必要があります。

そのパラメーターに指定された値やコンポーネント・バッファーで現在使用可能な情報を使用して、関数はコンポーネントから必要な情報を収集し、イベントをイベント表に挿入します。関数により、次が実行されます。

- PeopleSoft システム変数 `%Mode` を評価します。この変数には、イベントに関連した動詞を識別するための、`Add` (追加) および `Update` (更新)/`Display` (表示) などの値が格納されます。

- キー名を使用してコンポーネントから現行のキー値を収集し、ビジネス・オブジェクトのキー名と値のリストを作成します。
- PeopleCode 組み込み関数とメソッドを使用してイベント表に挿入する行に、すべての情報をロードします。

サンプルの PeopleCode 宣言および関数呼び出し

このセクションでは、前述の 2 つの関数のそれぞれについて、PeopleCode の宣言および関数呼び出しの例を示します。使用する関数に応じて、次のコード例のいずれかを PeopleCode エディターに挿入してください。

実際に関数を呼び出す前に、簡単な論理テストを使用してレコードまたはコンポーネントが実際に変更されていることを確認します。変更されていない場合、コネクタはパフォーマンスを向上させる関数を呼び出しません。また、%userid が CW でないことも確認します。これにより、コネクタが要求からのデータ変更を新しいアプリケーション・イベントと解釈しないようになります。

cw_publish_event のコード例:

```
/* Place this code in Component's SavePostChg() and define the four */
/* parameters used in the function call */

Declare Function cw_publish_event PeopleCode FUNCLIB_CW.CW_EVENT_NOT
    FieldFormula;
Component String &BONAME1;
Component String &KEYLIST1;
Component String &CW_PRIORITY1;
Component String &CONNID1;

    &BONAME1 = "Psft_Dept";
    &KEYLIST1 = "DEPT_TBL.SetId:DEPT_TBL.DeptId";
    &CW_PRIORITY1 = 2;
    &CONNID1 = "PeopleSoftConnector";

/* Check if Component Changed before calling function */
If ComponentChanged() and
    %userid <> "CW" then

    /* Publish this event to the IBM WebSphere
    CW_EVENT_TBL for polling */

    cw_publish_event(&BONAME1,&KEYLIST1,&CW_PRIORITY1,&CONNID1);

End-if;
```

cw_publish_future_dated_events のコード例:

```
/* Place this code in Component's SavePostChg() and define the four */
/* parameters used in the function call */

Declare Function cw_publish_future_dated_events PeopleCode FUNCLIB_CW.CW_EVENT_NOT
    FieldFormula;
Component String &BONAME1;
Component String &KEYLIST1;
Component String &CW_PRIORITY1;
Component String &CONNID1;
Component String &EFFDATE;

    &BONAME1 = "psft_CW_TEST_CI";
    &KEYLIST1 = "CW_PARENT_TBL.CW_PARENT_KEY1";
    &CW_PRIORITY1 = 2;
    &CONNID1 = "PeopleSoftConnector";
    &EFFDATE = CW_PARENT_TBL.CW_PARENT_DTTM";
```

```

/* Check if Component Changed before calling function */
If ComponentChanged() and
  %userid <> "CW" then

  /* Publish this event to the IBM WebSphere
  CW_EVENT_TBL for polling */

  cw_publish_future_dated_events(&BONAME1,&KEYLIST1,&CWPRRIORITY1,&CONNID1,
  &EFFDATE);

End-if;

```

イベントおよびアーカイブ表

コネクタはイベント表を使用して、イベントをポーリングし、取り出します。各イベントについて、コネクタはイベント表からビジネス・オブジェクトの名前、動詞、およびキーを取得します。また、この情報を使用してアプリケーションからエンティティ全体を取り出します。イベントが最初にログに記録された後でエンティティが変更されると、コネクタは初期イベントとその後のすべての変更を取得します。つまり、コネクタがイベント表からエンティティを取得する前にエンティティが変更および更新されると、コネクタは 1 回の取り出しで両方のデータ変更を取得します。

コネクタで処理された各イベントに関して、次の 3 種類の結果が発生する可能性があります。

- イベントは正常に処理されました。
- イベントは正常に処理されませんでした。
- イベントはサブスクライブされませんでした。

コネクタがイベントを取り出した後でイベント表からイベントが削除されない、と、不必要なスペースを占有することになります。しかし、削除されると、処理されなかったイベントはすべて失われ、イベント処理を監査できません。したがって、アダプターには、イベント表から削除されたイベントを格納するアーカイブ表が用意されています。

表 3 で、イベントおよびアーカイブ表の列について説明します。

表 3. イベントおよびアーカイブ表スキーマ

名前	説明	タイプ	制約
CW_EVENT_ID	イベントの内部 ID。PeopleSoft 内で生成された各イベントを識別する固有のキー・フィールド。	NUMBER	基本キー
CW_CONNECTOR_ID	イベントの宛先のコネクタの固有な ID。この値は、複数のコネクタが同じ表をポーリングする場合には重要です。	VARCHAR	

表3. イベントおよびアーカイブ表スキーマ (続き)

名前	説明	タイプ	制約
CW_OBJ_KEYS	ビジネス・オブジェクトのキーで、名前と値の対の形式で指定されます。関数呼び出し (cw_publish_events など) のパラメーターを定義する場合、名前はピリオドで区切った表名とフィールド名で構成されます。キーが複数の場合は、コロンまたは他の構成可能な区切り文字で区切ります (例えば、DEPT_TBL.SetId:DEPT_TBL.DeptId)。詳細については、126 ページの『EventKeyDelimiter』を参照してください。	VARCHAR	非ヌル
CW_OBJ	ビジネス・オブジェクトの名前。	VARCHAR	非ヌル
CW_VERB	イベントに関連した動詞。アダプターに組み込まれている PeopleCode 関数が、PeopleSoft で使用するシステム変数 %Mode に基づいて値 (Create、Retrieve、Update、Delete) を判別します。	VARCHAR	非ヌル
CW_PRIORITY	使用されません。		
CW_DTTM	イベントまたはアーカイブが発生した日付と時刻。	STRING	デフォルトの現在日付/時刻 (アーカイブ表の場合、実際のイベントの時間)。PeopleSoft では DATE フィールドのデータ型を STRING として処理し、また、これと同じ型で戻します。
CW_STATUS	-2 (統合ブローカーへのイベントの送信エラー) -1 (イベント処理エラー) 0 (ポーリング開始可能) 1 (統合ブローカーに送信済み) 2 (ビジネス・オブジェクトのサブスクリプションなし) 3 (進行中) この状況は、イベント表にのみ使用され、アーカイブ表には使用されません。 99 (将来の有効な日付のイベント) この状況は、イベント表にのみ使用され、アーカイブ表には使用されません。	VARCHAR	非ヌル

第 3 章 コネクターの構成

この章では、Connector Configurator について説明します。

- 『Connector Configurator の概要』
- 28 ページの『Connector Configurator の始動』
- 29 ページの『System Manager からの Configurator の実行』
- 29 ページの『コネクタ固有のプロパティ・テンプレートの作成』
- 32 ページの『新しい構成ファイルを作成』
- 33 ページの『既存ファイルの使用』
- 35 ページの『構成ファイルの完成』
- 35 ページの『構成ファイル・プロパティの設定』
- 42 ページの『構成ファイルの保管』
- 43 ページの『構成ファイルの変更』
- 43 ページの『構成の完了』
- 44 ページの『グローバル化環境における Connector Configurator の使用』

Connector Configurator の概要

Connector Configurator では、次の統合ブローカーで使用するアダプターのコネクタ・コンポーネントを構成できます。

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator、WebSphere MQ Integrator Broker、および WebSphere Business Integration Message Broker (WebSphere Message Brokers (WMQI) と総称)
- WebSphere Application Server (WAS)

Connector Configurator を使用して次の作業を行います。

- コネクタを構成するためのコネクタ固有のプロパティ・テンプレートを作成する。
- **コネクタ構成ファイル**を作成します。インストールするコネクタごとに構成ファイルを 1 つ作成する必要があります。
- 構成ファイル内のプロパティを設定する。
場合によっては、コネクタ・テンプレートでプロパティに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、ICS の場合はコラボレーションとともに使用するマップを指定し、必要に応じてメッセージング、ロギング、トレース、およびデータ・ハンドラー・パラメーターを指定する必要があります。

Connector Configurator の実行モードと使用する構成ファイルのタイプは、実行する統合ブローカーによって異なります。例えば、使用している統合ブローカーが

WMQI の場合、Connector Configurator を System Manager から実行するのではなく、直接実行します (『スタンドアロン・モードでの Configurator の実行』を参照)。

コネクタ構成プロパティには、標準の構成プロパティ (すべてのコネクタがもつプロパティ) と、コネクタ固有のプロパティ (特定のアプリケーションまたはテクノロジーのためにコネクタに必要なプロパティ) とが含まれます。

標準プロパティはすべてのコネクタにより使用されるので、標準プロパティを新規に定義する必要はありません。ファイルを作成すると、Connector Configurator により標準プロパティがこの構成ファイルに挿入されます。ただし、Connector Configurator で各標準プロパティの値を設定する必要があります。

注: このアダプターは、標準プロパティ Character Encoding を使用しません。また、このコネクタは国際化対応されているため、標準プロパティ Locale の値を変更できます。現在サポートされているロケールを確認するには、コネクタのリリース情報を参照してください。

標準プロパティの範囲は、ブローカーと構成によって異なる可能性があります。特定のプロパティに特定の値が設定されている場合にのみ使用できるプロパティがあります。Connector Configurator の「標準のプロパティ」ウィンドウには、特定の構成で設定可能なプロパティが表示されます。

ただし**コネクタ固有プロパティ**の場合は、最初にプロパティを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクタ固有プロパティのテンプレートを作成します。システム内ですでにテンプレートが作成されている場合には、作成されているテンプレートを使用します。システム内でまだテンプレートが作成されていない場合には、30 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

注: Connector Configurator は、Windows 環境内でのみ実行されます。UNIX 環境でコネクタを実行する場合には、Windows で Connector Configurator を使用して構成ファイルを変更し、このファイルを UNIX 環境へコピーします。

Connector Configurator の始動

以下の 2 種類のモードで Connector Configurator を開始および実行できます。

- スタンドアロン・モードで個別に実行
- System Manager から

スタンドアロン・モードでの Configurator の実行

どのブローカーを実行している場合にも、Connector Configurator を個別に実行し、コネクタ構成ファイルを編集できます。

これを行うには、以下のステップを実行します。

- 「スタート」>「プログラム」から、「IBM WebSphere InterChange Server」>「IBM WebSphere Business Integration Tools」>「Connector Configurator」をクリックします。
- 「ファイル」>「新規」>「コネクタ構成」を選択します。

- 「システム接続: Integration Broker」の隣のプルダウン・メニューをクリックします。使用しているブローカーに応じて、ICS、WebSphere Message Brokers、または WAS を選択します。

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (35 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

System Manager から Connector Configurator を実行できます。

Connector Configurator を実行するには、以下のステップを実行します。

1. System Manager を開きます。
2. 「System Manager」ウィンドウで、「統合コンポーネント・ライブラリー」アイコンを展開し、「コネクタ」を強調表示します。
3. System Manager メニュー・バーから、「ツール」>「**Connector Configurator**」をクリックします。「Connector Configurator」ウィンドウが開き、「新規コネクタ」ダイアログ・ボックスが表示されます。
4. 「システム接続: Integration Broker」の隣のプルダウン・メニューをクリックします。使用しているブローカーに応じて、ICS、WebSphere Message Brokers、または WAS を選択します。

既存の構成ファイルを編集するには、以下のステップを実行します。

- 「System Manager」ウィンドウの「コネクタ」フォルダーでいずれかの構成ファイルを選択し、右クリックします。Connector Configurator が開き、この構成ファイルの統合ブローカー・タイプおよびファイル名が上部に表示されます。
- Connector Configurator で「ファイル」>「開く」を選択します。プロジェクトまたはプロジェクトが保管されているディレクトリーからコネクタ構成ファイルを選択します。
- 「標準のプロパティ」タブをクリックし、この構成ファイルに含まれているプロパティを確認します。

コネクタ固有のプロパティ・テンプレートの作成

コネクタの構成ファイルを作成するには、コネクタ固有プロパティのテンプレートとシステム提供の標準プロパティが必要です。

コネクタ固有プロパティのテンプレートを新規に作成するか、または既存のコネクタ定義をテンプレートとして使用します。

- テンプレートの新規作成については、30 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でのこのテンプレートを保管します。既存のテンプレートは `¥WebSphereAdapters¥bin¥Data¥App` ディレクトリーにあります。

新規テンプレートの作成

このセクションでは、テンプレートでプロパティを作成し、プロパティの一般特性および値を定義し、プロパティ間の依存関係を指定する方法について説明します。次にそのテンプレートを保管し、新規コネクタ構成ファイルを作成するためのベースとして使用します。

Connector Configurator でテンプレートを作成するには、以下のステップを実行します。

1. 「ファイル」>「新規」>「コネクタ固有プロパティ・テンプレート」をクリックします。
2. 「コネクタ固有プロパティ・テンプレート」 ダイアログ・ボックスが表示されます。
 - 「新規テンプレート名を入力してください」の下の「名前」フィールドに、新規テンプレートの名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
 - テンプレートに含まれているコネクタ固有のプロパティ定義を調べるには、「テンプレート名」表示でそのテンプレートの名前を選択します。そのテンプレートに含まれているプロパティ定義のリストが「テンプレートのプレビュー」表示に表示されます。
3. テンプレートを作成するときには、ご使用のコネクタに必要なプロパティ定義に類似したプロパティ定義が含まれている既存のテンプレートを使用できます。ご使用のコネクタで使用するコネクタ固有のプロパティが表示されるテンプレートが見つからない場合は、自分で作成する必要があります。
 - 既存のテンプレートを変更する場合には、「変更する既存のテンプレートを選択してください: 検索テンプレート」の下の「テンプレート名」テーブルのリストから、テンプレート名を選択します。
 - このテーブルには、現在使用可能なすべてのテンプレートの名前が表示されます。テンプレートを検索することもできます。

一般特性の指定

「次へ」をクリックしてテンプレートを選択すると、「プロパティ: コネクタ固有プロパティ・テンプレート」ダイアログ・ボックスが表示されます。このダイアログ・ボックスには、定義済みプロパティの「一般」特性のタブと「値」の制限のタブがあります。「一般」表示には以下のフィールドがあります。

- **一般:**
 - プロパティ・タイプ
 - 更新されたメソッド
 - 説明
- **フラグ**
 - 標準フラグ
- **カスタム・フラグ**
 - フラグ

プロパティの一般特性の選択を終えたら、「値」タブをクリックします。

値の指定

「値」タブを使用すると、プロパティの最大長、最大複数値、デフォルト値、または値の範囲を設定できます。編集可能な値も許可されます。これを行うには、以下のステップを実行します。

1. 「値」タブをクリックします。「一般」のパネルに代わって「値」の表示パネルが表示されます。
2. 「プロパティを編集」表示でプロパティの名前を選択します。
3. 「最大長」および「最大複数値」のフィールドに値を入力します。

新規プロパティ値を作成するには、以下のステップを実行します。

1. 「プロパティを編集」リストでプロパティを選択し、右マウス・ボタンでクリックします。
2. ダイアログ・ボックスから「追加」を選択します。
3. 新規プロパティ値の名前を入力し、「OK」をクリックします。右側の「値」パネルに値が表示されます。

「値」パネルには、3つの列からなるテーブルが表示されます。

「値」の列には、「プロパティ値」ダイアログ・ボックスで入力した値と、以前に作成した値が表示されます。

「デフォルト値」の列では、値のいずれかをデフォルトとして指定することができます。

「値の範囲」の列には、「プロパティ値」ダイアログ・ボックスで入力した範囲が表示されます。

値が作成されて、グリッドに表示されると、そのテーブルの表示内から編集できるようになります。

テーブルにある既存の値の変更を行うには、その行の行番号をクリックして行全体を選択します。次に「値」フィールドを右マウス・ボタンでクリックし、「値の編集 (Edit Value)」をクリックします。

依存関係の設定

「一般」タブと「値」タブで変更を行ったら、「次へ」をクリックします。「依存関係: コネクター固有プロパティ・テンプレート」ダイアログ・ボックスが表示されます。

依存プロパティは、別のプロパティの値が特定の条件に合致する場合にのみ、テンプレートに組み込まれて、構成ファイルで使用されるプロパティです。例えば、テンプレートに `PollQuantity` が表示されるのは、トランスポート機構が `JMS` であり、`DuplicateEventElimination` が `True` に設定されている場合のみです。プロパティを依存プロパティとして指定し、依存する条件を設定するには、以下のステップを実行します。

1. 「使用可能なプロパティ」表示で、依存プロパティとして指定するプロパティを選択します。

2. 「プロパティを選択」フィールドで、ドロップダウン・メニューを使用して、条件値を持たせるプロパティを選択します。
3. 「条件演算子」フィールドで以下のいずれかを選択します。

== (等しい)

!= (等しくない)

> (より大)

< (より小)

>= (より大か等しい)

<= (より小か等しい)

4. 「条件値」フィールドで、依存プロパティをテンプレートに組み込むために必要な値を入力します。
5. 「使用可能なプロパティ」表示で依存プロパティを強調表示させて矢印をクリックし、「依存プロパティ」表示に移動させます。
6. 「完了」をクリックします。Connector Configurator により、XML 文書として入力した情報が、Connector Configurator がインストールされている %bin ディレクトリーの %data¥app の下に保管されます。

新しい構成ファイルを作成

構成ファイルを新規に作成するには、構成ファイルの名前を指定し、統合ブローカーを選択する必要があります。

- 「System Manager」ウィンドウで「コネクタ」フォルダーを右クリックし、「新規コネクタの作成」を選択します。Connector Configurator が開き、「新規コネクタ」ダイアログ・ボックスが表示されます。
- スタンドアロン・モードの場合は、Connector Configurator で「ファイル」>「新規」>「コネクタ構成」を選択します。「新規コネクタ」ウィンドウで、新規コネクタの名前を入力します。

また、統合ブローカーも選択する必要があります。選択したブローカーによって、構成ファイルに記述されるプロパティが決まります。ブローカーを選択するには、以下のステップを実行します。

- 「Integration Broker」フィールドで、ICS 接続、WebSphere Message Brokers 接続、WAS 接続のいずれかを選択します。
- この章で後述する説明に従って「新規コネクタ」ウィンドウの残りのフィールドに入力します。

コネクタ固有のテンプレートからの構成ファイルの作成

コネクタ固有のテンプレートを作成すると、テンプレートを使用して構成ファイルを作成できます。

1. 「ファイル」>「新規」>「コネクタ構成」をクリックします。
2. 以下のフィールドを含む「新規コネクタ」ダイアログ・ボックス表示されません。

- **名前**

コネクタの名前を入力します。名前では大文字と小文字が区別されます。入力する名前は、システムにインストールされているコネクタのファイル名に対応した一意の名前でなければなりません。

重要: Connector Configurator では、入力された名前のスペルはチェックされません。名前が正しいことを確認してください。

- **システム接続**

ICS 接続、WebSphere Message Brokers 接続、WAS のいずれかをクリックします。

- 「コネクタ固有プロパティ・テンプレート」を選択します。

ご使用のコネクタ用に設計したテンプレートの名前を入力します。「**テンプレート名**」表示に、使用可能なテンプレートが表示されます。「**テンプレート名**」表示で名前を選択すると、「**プロパティ・テンプレートのプレビュー**」表示に、そのテンプレートで定義されているコネクタ固有のプロパティが表示されます。

使用するテンプレートを選択し、「**OK**」をクリックします。

3. 構成しているコネクタの構成画面が表示されます。タイトル・バーに統合ブローカーとコネクタの名前が表示されます。ここですべてのフィールドに値を入力して定義を完了するか、ファイルを保管して後でフィールドに値を入力するかを選択できます。
4. ファイルを保管するには、「**ファイル**」>「**保管**」>「**ファイルに**」をクリックするか、「**ファイル**」>「**保管**」>「**プロジェクトに**」をクリックします。プロジェクトに保管するには、System Manager が実行中でなければなりません。ファイルとして保管する場合は、「**ファイル・コネクタを保管**」ダイアログ・ボックスが表示されます。*.cfg をファイル・タイプとして選択し、「**ファイル名**」フィールド内に名前が正しいスペル (大文字と小文字の区別を含む) で表示されていることを確認してから、ファイルを保管するディレクトリーにナビゲートし、「**保管**」をクリックします。Connector Configurator のメッセージ・パネルの状況表示に、構成ファイルが正常に作成されたことが示されます。

重要: ここで設定するディレクトリー・パスおよび名前は、コネクタの始動ファイルで指定するコネクタ構成ファイルのパスおよび名前に一致している必要があります。

5. この章で後述する手順に従って、「Connector Configurator」ウィンドウの各タブにあるフィールドに値を入力し、コネクタ定義を完了します。

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

- **コネクタ定義ファイル。**

コネクタ定義ファイルは、特定のコネクタのプロパティと、適用可能なデフォルト値がリストされたテキスト・ファイルです。コネクタの配布パッケージ

ジの `¥repository` ディレクトリー内には、このようなファイルが格納されていることがあります (通常、このファイルの拡張子は `.txt` です。例えば、XML コネクタの場合は `CN_XML.txt` です)。

- ICS リポジトリー・ファイル。
コネクタの以前の ICS 実装で使用した定義は、そのコネクタの構成で使用されたリポジトリー・ファイルで使用可能になります。そのようなファイルの拡張子は、通常 `.in` または `.out` です。
- コネクタの以前の構成ファイル。
これらのファイルの拡張子は、通常 `*.cfg` です。

これらのいずれのファイル・ソースにも、コネクタのコネクタ固有プロパティのほとんど、あるいはすべてが含まれますが、この章内の後で説明するように、コネクタ構成ファイルは、ファイルを開いて、プロパティを設定しない限り完成しません。

既存ファイルを使用してコネクタを構成するには、Connector Configurator でそのファイルを開き、構成を修正し、そのファイルを再度保管する必要があります。

以下のステップを実行して、ディレクトリーから `*.txt`、`*.cfg`、または `*.in` ファイルを開きます。

1. Connector Configurator 内で、「ファイル」>「開く」>「ファイルから」をクリックします。
2. 「ファイル・コネクタを開く」ダイアログ・ボックス内で、以下のいずれかのファイル・タイプを選択して、使用可能なファイルを調べます。
 - 構成 (`*.cfg`)
 - ICS リポジトリー (`*.in`、`*.out`)

ICS 環境でのコネクタの構成にリポジトリー・ファイルが使用された場合には、このオプションを選択します。リポジトリー・ファイルに複数のコネクタ定義が含まれている場合は、ファイルを開くとすべての定義が表示されます。

- すべてのファイル (`*.*`)

コネクタのアダプター・パッケージに `*.txt` ファイルが付属していた場合、または別の拡張子で定義ファイルが使用可能である場合は、このオプションを選択します。

3. ディレクトリー表示内で、適切なコネクタ定義ファイルへ移動し、ファイルを選択し、「開く」をクリックします。

System Manager プロジェクトからコネクタ構成を開くには、以下のステップを実行します。

1. System Manager を始動します。System Manager が開始されている場合にのみ、構成を System Manager から開いたり、System Manager に保管したりできます。
2. Connector Configurator を始動します。
3. 「ファイル」>「開く」>「プロジェクトから」をクリックします。

構成ファイルの完成

構成ファイルを開くか、プロジェクトからコネクターを開くと、「Connector Configurator」ウィンドウに構成画面が表示されます。この画面には、現在の属性と値が表示されます。

構成画面のタイトルには、ファイル内で指定された統合ブローカーとコネクターの名前が表示されます。正しいブローカーが設定されていることを確認してください。正しいブローカーが設定されていない場合、コネクターを構成する前にブローカー値を変更してください。これを行うには、以下のステップを実行します。

1. 「標準のプロパティ」タブで、BrokerType プロパティの値フィールドを選択します。ドロップダウン・メニューで、値 ICS、WMQI、または WAS を選択します。
2. 選択したブローカーに関連付けられているプロパティが「標準のプロパティ」タブに表示されます。ここでファイルを保管するか、または 38 ページの『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。
3. 構成が完了したら、「ファイル」>「保管」>「プロジェクトに」を選択するか、または「ファイル」>「保管」>「ファイルに」を選択します。

ファイルに保管する場合は、*.cfg を拡張子として選択し、ファイルの正しい格納場所を選択して、「保管」をクリックします。

複数のコネクター構成を開いている場合、構成をすべてファイルに保管するには「すべてファイルに保管」を選択し、コネクター構成をすべて System Manager プロジェクトに保管するには「すべてプロジェクトに保管」をクリックします。

Connector Configurator では、ファイルを保管する前に、必須の標準プロパティすべてに値が設定されているかどうかを確認されます。必須の標準プロパティに値が設定されていない場合、Connector Configurator は、検証が失敗したというメッセージを表示します。構成ファイルを保管するには、そのプロパティの値を指定する必要があります。

構成ファイル・プロパティの設定

新規のコネクター構成ファイルを作成して名前を付けるとき、または既存のコネクター構成ファイルを開くときには、Connector Configurator によって構成画面が表示されます。構成画面には、必要な構成値のカテゴリーに対応する複数のタブがあります。

Connector Configurator では、すべてのブローカーで実行されているコネクターで、以下のカテゴリーのプロパティに値が設定されている必要があります。

- 標準のプロパティ
- コネクター固有のプロパティ
- サポートされるビジネス・オブジェクト
- トレース/ログ・ファイルの値
- データ・ハンドラー (保証付きイベント・デリバリーで JMS メッセージングを使用するコネクターの場合に該当する)

注: JMS メッセージングを使用するコネクタの場合は、データをビジネス・オブジェクトに変換するデータ・ハンドラーの構成に関して追加のカテゴリーが表示される場合があります。

ICS で実行されているコネクタの場合、以下のプロパティの値も設定されている必要があります。

- 関連マップ
- リソース
- メッセージング (該当する場合)

重要: Connector Configurator では、英語文字セットまたは英語以外の文字セットのいずれのプロパティ値も設定可能です。ただし、標準のプロパティおよびコネクタ固有プロパティ、およびサポートされるビジネス・オブジェクトの名前では、英語文字セットのみを使用する必要があります。

標準プロパティとコネクタ固有プロパティの違いは、以下のとおりです。

- コネクタの標準プロパティは、コネクタのアプリケーション固有のコンポーネントとブローカー・コンポーネントの両方によって共有されます。すべてのコネクタが同じ標準プロパティのセットを使用します。これらのプロパティの説明は、各アダプター・ガイドの付録 A にあります。変更できるのはこれらの値の一部のみです。
- アプリケーション固有のプロパティは、コネクタのアプリケーション固有コンポーネント (アプリケーションと直接対話するコンポーネント) のみに適用されます。各コネクタには、そのコネクタのアプリケーションだけで使用されるアプリケーション固有のプロパティがあります。これらのプロパティには、デフォルト値が用意されているものもあれば、そうでないものもあります。また、一部のデフォルト値は変更することができます。各アダプター・ガイドのインストールおよび構成の章に、アプリケーション固有のプロパティおよび推奨値が記述されています。

「標準プロパティ」と「コネクタ固有プロパティ」のフィールドは、どのフィールドが構成可能であるかを示すために色分けされています。

- 背景がグレーのフィールドは、標準のプロパティを表します。値を変更することはできますが、名前の変更およびプロパティの除去はできません。
- 背景が白のフィールドは、アプリケーション固有のプロパティを表します。これらのプロパティは、アプリケーションまたはコネクタの特定のニーズによって異なります。値の変更も、これらのプロパティの除去も可能です。
- 「値」フィールドは構成できます。
- プロパティごとに「更新メソッド」フィールドが表示されます。これは、変更された値をアクティブにするためにコンポーネントまたはエージェントの再始動が必要かどうかを示します。この設定を構成することはできません。

標準コネクタ・プロパティの設定

標準のプロパティの値を変更するには、以下の手順を実行します。

1. 値を設定するフィールド内でクリックします。

2. 値を入力するか、ドロップダウン・メニューが表示された場合にはメニューから値を選択します。
3. 標準のプロパティの値をすべて入力後、以下のいずれかを実行することができます。
 - 変更内容を破棄し、元の値を保持したままで Connector Configurator を終了するには、「ファイル」>「終了」をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら「いいえ」をクリックします。
 - Connector Configurator 内の他のカテゴリーの値を入力するには、そのカテゴリーのタブを選択します。「標準のプロパティ」(またはその他のカテゴリー) で入力した値は、次のカテゴリーに移動しても保持されます。ウィンドウを閉じると、すべてのカテゴリーで入力した値を一括して保管するかまたは破棄するかを確認するプロンプトが出されます。
 - 修正した値を保管するには、「ファイル」>「終了」をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら「はい」をクリックします。「ファイル」メニューまたはツールバーから「保管」>「ファイルに」をクリックする方法もあります。

アプリケーション固有の構成プロパティの設定

アプリケーション固有の構成プロパティの場合、プロパティ名の追加または変更、値の構成、プロパティの削除、およびプロパティの暗号化が可能です。プロパティのデフォルトの長さは 255 文字です。

1. グリッドの左上端の部分で右マウス・ボタンをクリックします。ポップアップ・メニュー・バーが表示されます。プロパティを追加するときは「追加」をクリックします。子プロパティを追加するには、親の行番号で右マウス・ボタンをクリックし、「子を追加」をクリックします。
2. プロパティまたは子プロパティの値を入力します。
3. プロパティを暗号化するには、「暗号化」ボックスを選択します。
4. 36 ページの『標準コネクタ・プロパティの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

各プロパティごとに表示される「更新メソッド」は、変更された値をアクティブにするためにコンポーネントまたはエージェントの再始動が必要かどうかを示します。

重要: 事前設定のアプリケーション固有のコネクタ・プロパティ名を変更すると、コネクタに障害が発生する可能性があります。コネクタをアプリケーションに接続したり正常に実行したりするために、特定のプロパティ名が必要である場合があります。

コネクタ・プロパティの暗号化

「コネクタ固有プロパティ」ウィンドウの「暗号化」チェック・ボックスにチェックマークを付けると、アプリケーション固有のプロパティを暗号化することができます。値の暗号化を解除するには、「暗号化」チェック・ボックスをクリックしてチェックマークを外し、「検証」ダイアログ・ボックスに正しい値を入力し、「OK」をクリックします。入力された値が正しい場合は、暗号化解除された値が表示されます。

各プロパティとそのデフォルト値のリストおよび説明は、各コネクターのアダプター・ユーザーズ・ガイドにあります。

プロパティに複数の値がある場合には、プロパティの最初の値に「暗号化」チェック・ボックスが表示されます。「暗号化」を選択すると、そのプロパティのすべての値が暗号化されます。プロパティの複数の値を暗号化解除するには、そのプロパティの最初の値の「暗号化」チェック・ボックスをクリックしてチェックマークを外してから、「検証」ダイアログ・ボックスで新規の値を入力します。入力値が一致すれば、すべての複数值が暗号化解除されます。

更新メソッド

付録 A『コネクターの標準構成プロパティ』の 106 ページの『プロパティ値の設定と更新』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

コネクターで使用するビジネス・オブジェクトを指定するには、Connector Configurator の「サポートされているビジネス・オブジェクト」タブを使用します。汎用ビジネス・オブジェクトと、アプリケーション固有のビジネス・オブジェクトの両方を指定する必要があり、またそれらのビジネス・オブジェクト間のマップの関連を指定することが必要です。

注: コネクターによっては、アプリケーションでイベント通知や (メタオブジェクトを使用した) 追加の構成を実行するために、特定のビジネス・オブジェクトをサポートされているものとして指定することが必要な場合もあります。詳細は、「コネクター開発ガイド (C++ 用)」または「コネクター開発ガイド (Java 用)」を参照してください。

ご使用のブローカーが ICS の場合

ビジネス・オブジェクト定義がコネクターでサポートされることを指定する場合や、既存のビジネス・オブジェクト定義のサポート設定を変更する場合は、「サポートされているビジネス・オブジェクト」タブをクリックし、以下のフィールドを使用してください。

ビジネス・オブジェクト名: ビジネス・オブジェクト定義がコネクターによってサポートされることを指定するには、System Manager を実行し、以下の手順を実行します。

1. 「ビジネス・オブジェクト名」リストで空のフィールドをクリックします。
System Manager プロジェクトに存在するすべてのビジネス・オブジェクト定義を示すドロップダウン・リストが表示されます。
2. 追加するビジネス・オブジェクトをクリックします。
3. ビジネス・オブジェクトの「エージェント・サポート」(以下で説明) を設定します。
4. 「Connector Configurator」ウィンドウの「ファイル」メニューで、「プロジェクトに保管」をクリックします。追加したビジネス・オブジェクト定義に指定されたサポートを含む、変更されたコネクター定義が、System Manager の ICL (Integration Component Library) プロジェクトに保管されます。

サポートされるリストからビジネス・オブジェクトを削除する場合は、以下の手順を実行します。

1. ビジネス・オブジェクト・フィールドを選択するため、そのビジネス・オブジェクトの左側の番号をクリックします。
2. 「Connector Configurator」ウィンドウの「編集」メニューから、「行を削除」をクリックします。リスト表示からビジネス・オブジェクトが除去されます。
3. 「ファイル」メニューから、「プロジェクトの保管」をクリックします。

サポートされるリストからビジネス・オブジェクトを削除すると、コネクタ定義が変更され、削除されたビジネス・オブジェクトはコネクタのこの実装で使用不可になります。コネクタのコードに影響したり、そのビジネス・オブジェクト定義そのものが System Manager から削除されることはありません。

エージェント・サポート: ビジネス・オブジェクトがエージェント・サポートを備えている場合、システムは、コネクタ・エージェントを介してアプリケーションにデータを配布する際にそのビジネス・オブジェクトの使用を試みます。

一般に、コネクタのアプリケーション固有ビジネス・オブジェクトは、そのコネクタのエージェントによってサポートされますが、汎用ビジネス・オブジェクトはサポートされません。

ビジネス・オブジェクトがコネクタ・エージェントによってサポートされるよう指定するには、「エージェント・サポート」ボックスにチェックマークを付けます。「Connector Configurator」ウィンドウでは「エージェント・サポート」の選択の妥当性は検査されません。

最大トランザクション・レベル: コネクタの最大トランザクション・レベルは、そのコネクタがサポートする最大のトランザクション・レベルです。

ほとんどのコネクタの場合、選択可能な項目は「最大限の努力」のみです。

トランザクション・レベルの変更を有効にするには、サーバーを再始動する必要があります。

ご使用のブローカーが WebSphere Message Broker の場合

スタンドアロン・モードで作業している (System Manager に接続していない) 場合、手動でビジネス・オブジェクト名を入力する必要があります。

System Manager を実行している場合、「サポートされているビジネス・オブジェクト」タブの「ビジネス・オブジェクト名」列の下にある空のボックスを選択できます。コンボ・ボックスが表示され、コネクタが属する統合コンポーネント・ライブラリー・プロジェクトから選択可能なビジネス・オブジェクトのリストが示されます。リストから必要なビジネス・オブジェクトを選択します。

「メッセージ・セット ID」は、WebSphere Business Integration Message Broker 5.0 のオプションのフィールドです。この ID が提供される場合、一意である必要はありません。ただし、WebSphere MQ Integrator および Integrator Broker 2.1 の場合は、一意の ID を提供する必要があります。

ご使用のブローカーが WAS の場合

使用するブローカー・タイプとして WebSphere Application Server を選択した場合、Connector Configurator にメッセージ・セット ID は必要ありません。「サポートされているビジネス・オブジェクト」タブには、サポートされるビジネス・オブジェクトの「ビジネス・オブジェクト名」列のみが表示されます。

スタンドアロン・モードで作業している (System Manager に接続していない) 場合、手動でビジネス・オブジェクト名を入力する必要があります。

System Manager を実行している場合、「サポートされているビジネス・オブジェクト」タブの「ビジネス・オブジェクト名」列の下にある空のボックスを選択できます。コンボ・ボックスが表示され、コネクターが属する統合コンポーネント・ライブラリー・プロジェクトから選択可能なビジネス・オブジェクトのリストが示されます。このリストから必要なビジネス・オブジェクトを選択します。

関係付けられたマップ (ICS のみ)

各コネクターは、現在 WebSphere InterChange Server でアクティブなビジネス・オブジェクト定義、およびそれらの関連付けられたマップのリストをサポートします。このリストは、「関連付けられたマップ」タブを選択すると表示されます。

ビジネス・オブジェクトのリストには、エージェントでサポートされるアプリケーション固有のビジネス・オブジェクトと、コントローラーがサブスクリプション・コラボレーションに送信する、対応する汎用オブジェクトが含まれます。マップの関連によって、アプリケーション固有のビジネス・オブジェクトを汎用ビジネス・オブジェクトに変換したり、汎用ビジネス・オブジェクトをアプリケーション固有のビジネス・オブジェクトに変換したりするとき、どのマップを使用するかが決定されます。

特定のソースおよび宛先ビジネス・オブジェクトについて一意的に定義されたマップを使用する場合、表示を開くと、マップは常にそれらの該当するビジネス・オブジェクトに関連付けられます。ユーザーがそれらを変更する必要はありません (変更できません)。

サポートされるビジネス・オブジェクトで使用可能なマップが複数ある場合は、そのビジネス・オブジェクトを、使用する必要のあるマップに明示的にバインドすることが必要になります。

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

これらは、「サポートされているビジネス・オブジェクト」タブで指定した、このコネクターでサポートされるビジネス・オブジェクトです。「サポートされているビジネス・オブジェクト」タブでビジネス・オブジェクトを追加指定した場合、その内容は、「Connector Configurator」ウィンドウの「ファイル」メニューから「プロジェクトに保管」を選択して、変更を保管した後に、このリストに反映されます。

- **関連付けられたマップ**

この表示には、コネクターの、サポートされるビジネス・オブジェクトでの使用のためにシステムにインストールされたすべてのマップが表示されます。各マップのソース・ビジネス・オブジェクトは、「**ビジネス・オブジェクト名**」表示でマップ名の左側に表示されます。

- **明示的**

場合によっては、関連マップを明示的にバインドすることが必要になります。

明示的バインディングが必要なのは、特定のサポートされるビジネス・オブジェクトに複数のマップが存在する場合のみです。ICS は、ブート時、各コネクターでサポートされるそれぞれのビジネス・オブジェクトにマップを自動的にバインドしようとします。複数のマップでその入力データとして同一のビジネス・オブジェクトが使用されている場合、サーバーは、他のマップのスーパーセットである 1 つのマップを見つけて、バインドしようとします。

他のマップのスーパーセットであるマップがないと、サーバーは、ビジネス・オブジェクトを単一のマップにバインドすることができないため、バインディングを明示的に設定することが必要になります。

以下の手順を実行して、マップを明示的にバインドします。

1. 「**明示的 (Explicit)**」列で、バインドするマップのチェック・ボックスにチェックマークを付けます。
2. ビジネス・オブジェクトに関連付けるマップを選択します。
3. 「Connector Configurator」ウィンドウの「**ファイル**」メニューで、「**プロジェクトに保管**」をクリックします。
4. プロジェクトを ICS に配置します。
5. 変更を有効にするため、サーバーをリブートします。

リソース (ICS)

「リソース」タブでは、コネクター・エージェントが、コネクター・エージェント並列処理を使用して同時に複数のプロセスを処理するかどうか、またどの程度処理するかを決定する値を設定できます。

すべてのコネクターがこの機能をサポートしているわけではありません。複数のプロセスを使用するよりも複数のスレッドを使用する方が通常は効率的であるため、Java でマルチスレッドとして設計されたコネクター・エージェントを実行している場合、この機能を使用することはお勧めできません。

メッセージング (ICS)

メッセージング・プロパティは、DeliveryTransport 標準プロパティの値として MQ を設定し、ブローカー・タイプとして ICS を設定した場合にのみ、使用可能です。これらのプロパティは、コネクターによるキューの使用方法に影響します。

トレース/ログ・ファイル値の設定

コネクタ構成ファイルまたはコネクタ定義ファイルを開くと、Connector Configurator は、そのファイルのログおよびトレースの値をデフォルト値として使用します。Connector Configurator 内でこれらの値を変更できます。

ログとトレースの値を変更するには、以下の手順を実行します。

1. 「トレース/ログ・ファイル」タブをクリックします。
2. ログとトレースのどちらでも、以下のいずれかまたは両方へのメッセージの書き込みを選択できます。

- コンソールに (STDOUT):
ログ・メッセージまたはトレース・メッセージを STDOUT ディスプレイに書き込みます。

注: STDOUT オプションは、Windows プラットフォームで実行しているコネクタの「トレース/ログ・ファイル」タブでのみ使用できます。

- ファイルに:
ログ・メッセージまたはトレース・メッセージを指定されたファイルに書き込みます。ファイルを指定するには、ディレクトリー・ボタン (省略符号) をクリックし、指定する格納場所へ移動し、ファイル名を指定し、「保管」をクリックします。ログ・メッセージまたはトレース・メッセージは、指定した場所の指定したファイルに書き込まれます。

注: ログ・ファイルとトレース・ファイルはどちらも単純なテキスト・ファイルです。任意のファイル拡張子を使用してこれらのファイル名を設定できます。ただし、トレース・ファイルの場合、拡張子として .trc ではなく .trace を使用することをお勧めします。これは、システム内に存在する可能性がある他のファイルとの混同を避けるためです。ログ・ファイルの場合、通常使用されるファイル拡張子は .log および .txt です。

データ・ハンドラー

データ・ハンドラー・セクションの構成が使用可能となるのは、DeliveryTransport の値に JMS を、また ContainerManagedEvents の値に JMS を指定した場合のみです。すべてのアダプターでデータ・ハンドラーを使用できるわけではありません。

これらのプロパティに使用する値については、『付録 A. コネクタの標準構成プロパティ』にある ContainerManagedEvents の下の説明を参照してください。その他の詳細は、「コネクタ開発ガイド (C++ 用)」または「コネクタ開発ガイド (Java 用)」を参照してください。

構成ファイルの保管

コネクタの構成が完了したら、コネクタ構成ファイルを保管します。Connector Configurator では、構成中に選択したブローカー・モードでファイルを保管します。Connector Configurator のタイトル・バーには現在のブローカー・モード (ICS、WMQI、または WAS) が常に表示されます。

ファイルは XML 文書として保管されます。XML 文書は次の 3 通りの方法で保管できます。

- System Manager から、統合コンポーネント・ライブラリーに *.con 拡張子付きファイルとして保管します。
- System Manager から、指定したディレクトリーに *.con 拡張子付きファイルとして保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに *.cfg 拡張子付きファイルとして保管します。デフォルトでは、このファイルは %WebSphereAdapters%bin\Data¥App に保管されます。
- WebSphere Application Server プロジェクトをセットアップしている場合には、このファイルを WebSphere Application Server プロジェクトに保管することもできます。

System Manager でのプロジェクトの使用法、および配置の詳細については、以下のガイドを参照してください。

- ICS: 「*WebSphere InterChange Server* システム・インプリメンテーション・ガイド」
- WebSphere Message Brokers: 「*WebSphere Message Brokers* 使用アダプター・インプリメンテーション・ガイド」
- WAS: 「アダプター実装ガイド (*WebSphere Application Server*)」

構成ファイルの変更

既存の構成ファイルの統合ブローカー設定を変更できます。これにより、他のブローカーで使用する構成ファイルを新規に作成するときに、このファイルをテンプレートとして使用できます。

注: 統合ブローカーを切り替える場合には、ブローカー・モード・プロパティーと同様に他の構成プロパティーも変更する必要があります。

既存の構成ファイルでのブローカーの選択を変更するには、以下の手順を実行します (オプション)。

- Connector Configurator で既存の構成ファイルを開きます。
- 「標準のプロパティー」タブを選択します。
- 「標準のプロパティー」タブの「**BrokerType**」フィールドで、ご使用のブローカーに合った値を選択します。
現行値を変更すると、プロパティー画面の利用可能なタブおよびフィールド選択がただちに更新され、選択した新規ブローカーに適したタブとフィールドのみが表示されます。

構成の完了

コネクターの構成ファイルを作成し、そのファイルを変更した後で、コネクターの始動時にコネクターが構成ファイルの位置を特定できるかどうかを確認してください。

これを行うには、コネクタが使用する始動ファイルを開き、コネクタ構成ファイルに使用されている格納場所とファイル名が、ファイルに対して指定した名前およびファイルを格納したディレクトリまたはパスと正確に一致しているかどうかを検証します。

グローバル化環境における Connector Configurator の使用

Connector Configurator はグローバル化され、構成ファイルと統合ブローカー間の文字変換を処理できます。Connector Configurator では、ネイティブなエンコード方式を使用しています。構成ファイルに書き込む場合は UTF-8 エンコード方式を使用します。

Connector Configurator は、以下の場所で英語以外の文字をサポートします。

- すべての値のフィールド
- ログ・ファイルおよびトレース・ファイル・パス（「トレース/ログ・ファイル」タブで指定）

CharacterEncoding および Locale 標準構成プロパティのドロップ・リストに表示されるのは、サポートされる値の一部のみです。ドロップ・リストに、サポートされる他の値を追加するには、製品ディレクトリの %Data%Std%stdConnProps.xml ファイルを手動で変更する必要があります。

例えば、Locale プロパティの値のリストにロケール en_GB を追加するには、stdConnProps.xml ファイルを開き、以下に太文字で示した行を追加してください。

```
<Property name="Locale"
isRequired="true"
updateMethod="component restart">
  <ValidType>String</ValidType>
  <ValidValues>
    <Value>ja_JP</Value>
    <Value>ko_KR</Value>
    <Value>zh_CN</Value>
    <Value>zh_TW</Value>
    <Value>fr_FR</Value>
    <Value>de_DE</Value>
    <Value>it_IT</Value>
    <Value>es_ES</Value>
    <Value>pt_BR</Value>
    <Value>en_US</Value>
    <Value>en_GB</Value>
  <DefaultValue>en_US</DefaultValue>
</ValidValues>
</Property>
```

第 4 章 コネクタ用のビジネス・オブジェクトについて

この章は、以下のセクションから構成されています。

- 『ビジネス・オブジェクトおよび属性の命名規則』
- 46 ページの『ビジネス・オブジェクトの構造』
- 51 ページの『ビジネス・オブジェクトの作成』
- 55 ページの『ビジネス・オブジェクト動詞の処理』
- 64 ページの『ビジネス・オブジェクトの属性プロパティ』
- 66 ページの『ビジネス・オブジェクトのアプリケーション固有情報』

IBM WebSphere Business Integration Adapter for PeopleSoft のコネクタ・コンポーネントはメタデータ主導型です。WebSphere Business Integration システムでは、**メタデータ**はアプリケーション固有のデータで、WebSphere ビジネス・オブジェクトに格納され、コネクタのアプリケーションとの対話機能を支援します。メタデータ主導型コネクタは、サポートする各ビジネス・オブジェクトを、コネクタ内にハードコーディングされた命令ではなく、ビジネス・オブジェクト定義にエンコードされたメタデータに基づいて処理します。

ビジネス・オブジェクトのメタデータには、ビジネス・オブジェクトの構造、属性プロパティの設定、およびアプリケーション固有の情報の内容が含まれます。コネクタはメタデータ主導型のため、コネクタ・コードを変更せずに新規ビジネス・オブジェクトや変更されたビジネス・オブジェクトを処理できます。コネクタはビジネス・オブジェクト定義とそのアプリケーション固有情報を使用して、コンポーネント・インターフェースのデータを見つけて操作します。

コネクタには、サポート・ビジネス・オブジェクトの構造、親と子ビジネス・オブジェクト間の関係、アプリケーション固有情報のフォーマット、および対話に使用するコンポーネント・インターフェースについて、前提事項があります。したがって、コネクタが処理するビジネス・オブジェクトを作成または変更する場合は、コネクタが従うように指定されているルールに適合する必要があります。適合しないと、コネクタは新規または変更されたビジネス・オブジェクトを正しく処理できません。

この章では、コネクタによるビジネス・オブジェクトの処理方法と、コネクタの前提事項について説明します。この情報は、新しいビジネス・オブジェクトの実装についての推奨、または既存のビジネス・オブジェクトの変更ガイドとして使用できます。

ビジネス・オブジェクトおよび属性の命名規則

分かりやすくするために、PeopleSoft コンポーネント・インターフェース内のデータを表すためにビジネス・オブジェクトに名前を付ける場合、プレフィックス PSFT_を使用します。すべての子ビジネス・オブジェクトもトップレベルのビジネス・オブジェクトもこの規則に従います。例えば、EMERGENCY_CNTCT 表内の親データと

EMERGENCY_PHONE 表内の子データを表すビジネス・オブジェクトを作成する場合、対応するビジネス・オブジェクトの名前をそれぞれ、PSFT_EmergencyContact と PSFT_EmergencyPhone とします。

注: 本マニュアルでは、**階層**ビジネス・オブジェクトとは、そのビジネス・オブジェクトがすべてのレベルで含むすべての子ビジネス・オブジェクトを含めた、完全なビジネス・オブジェクトを指します。**個々の**ビジネス・オブジェクトとは、所有する子ビジネス・オブジェクトとは別個の、単一のビジネス・オブジェクトを指します。**トップレベルの**ビジネス・オブジェクトとは、それ自身は親ビジネス・オブジェクトを持たない階層構造の最上位の個々のビジネス・オブジェクトを指します。

ビジネス・オブジェクトの構造

PeopleSoft データを表す WebSphere ビジネス・オブジェクトは、フラットまたは階層構造のどちらでも可能です。**フラット**なビジネス・オブジェクトの属性はすべて単純です。つまり、単一の値 (ストリングまたは整数) を表します。

階層ビジネス・オブジェクトは、単純属性に加え、単一カーディナリティーの子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を表す属性を持つことができます。さらに今度は、これらの各ビジネス・オブジェクトが単一カーディナリティーの子ビジネス・オブジェクトやビジネス・オブジェクトの配列などを持つことができます。

単一カーディナリティー関係は、親ビジネス・オブジェクト内の属性が単一の子ビジネス・オブジェクトを表す場合に発生します。この場合、子ビジネス・オブジェクトは 1 つのレコードのみを持つことができる PeopleSoft コレクションを表します。属性タイプは、子ビジネス・オブジェクトのタイプと同じです。

複数カーディナリティー関係は、親ビジネス・オブジェクト内の属性が子ビジネス・オブジェクトの配列を表す場合に発生します。この場合、子ビジネス・オブジェクトは複数のレコードを持つことができる PeopleSoft コレクションを表します。属性タイプは、子ビジネス・オブジェクトの配列のタイプと同じです。

コネクタは、すべての階層ビジネス・オブジェクトが単一 PeopleSoft コンポーネント・インターフェースを表すことを前提としています。階層ビジネス・オブジェクトは複数の PeopleSoft レコード内のデータを表すことができますが、コネクタは階層内の子ビジネス・オブジェクトそれぞれがコンポーネント・インターフェース内の単一コレクションを表すことを前提としています。コネクタは PeopleSoft の (基本となるデータベースではなく) コンポーネント・アーキテクチャーと処理機能を使用して、コンポーネント・レベルで定義されるビジネス・ロジックを活用します。

ビジネス・オブジェクトを定義する際、PeopleSoft コレクションが、対応する個々のビジネス・オブジェクトが持つ単純属性よりも多くのフィールドを持つ可能性があります (つまり、コレクションの一部のフィールドは、ビジネス・オブジェクトで表されません)。作成時に、ビジネス・オブジェクトの処理に必要なフィールドのみを組み込みます。

コンポーネント・インターフェースとビジネス・オブジェクトの関係

コネクタには、処理する各ビジネス・オブジェクトごとに PeopleSoft ビジネス・コンポーネントとコンポーネント・インターフェースが必要ですが、WebSphere Business Integration システムには、ビジネス・コンポーネント、コンポーネント・インターフェース、PeopleSoft 固有のビジネス・オブジェクトのいずれも用意されていないので、コネクタを使用するためにはこれらのオブジェクトを作成する必要があります。

注: 可搬性のために、コネクタの作成はすべて 1 つの PeopleSoft プロジェクトで行ってください。

前述の必要なオブジェクトを作成すると、Application Designer を使用して、コネクタがサポート・ビジネス・オブジェクトを処理するのに必要なクラス構造を作成できます。詳細については、52 ページの『API の生成』を参照してください。

これらのオブジェクトの作成を支援するために、このセクションでは次について説明します。

- 『コンポーネントとコンポーネント・インターフェースの例』
- 51 ページの『ビジネス・オブジェクトの例』
- 51 ページの『ビジネス・オブジェクトの作成』

コンポーネントとコンポーネント・インターフェースの例

図 5 に、2 つのページと 3 つのレコードを持つ単純なコンポーネント EMER_CONTACT を示します。示されているページ (PERSONAL_DATA_PANEL1) には、各従業員の緊急連絡先情報が格納されています。

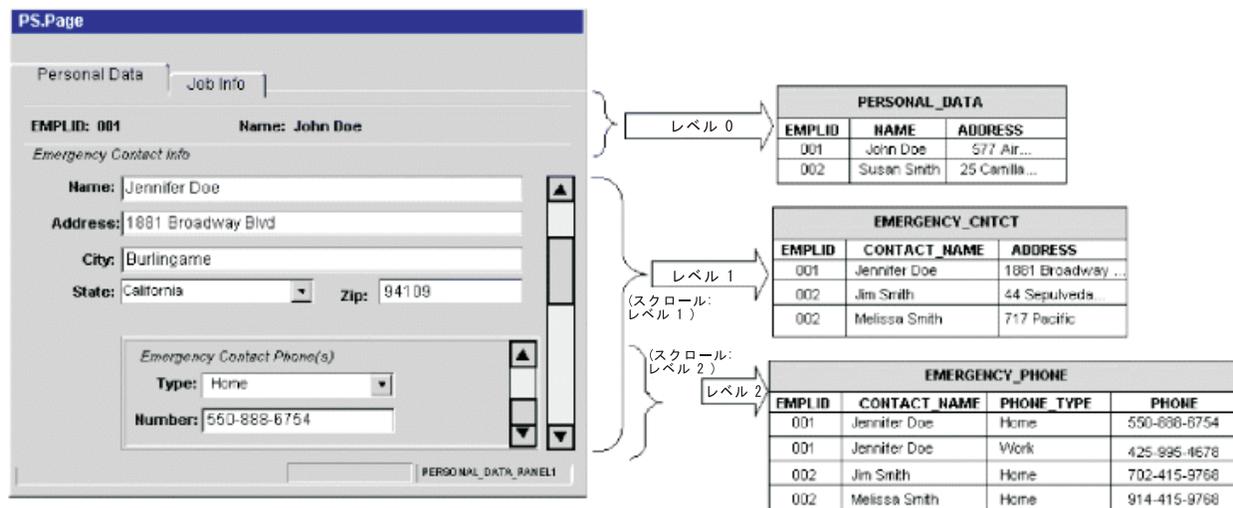


図 5. コンポーネントの例

前述の図は、個人データ・ページの各レベルで表示されるデータと、それを格納する対応するレコードとの関係を示します。

- レベル 0 のデータは PERSONAL_DATA レコードに格納され、そのキー・フィールドは EMPLID です。

- レベル 1 のデータは EMERGENCY_CNTCT レコードに格納されます。各従業員には複数の緊急連絡先がある可能性があるため、各連絡先は名前 (CONTACT_NAME) と従業員 ID (EMPLID) で一意に識別されます。レベル 1 のスクロールにより、連絡先コレクション全体のデータを表示できます。
- レベル 2 のデータは EMERGENCY_PHONE レコードに格納されます。各連絡先には複数の電話がある可能性があるため、各電話番号は種類 (PHONE_TYPE)、連絡先の名前 (CONTACT_NAME)、および従業員 ID (EMPLID) で一意に識別されます。レベル 2 のスクロールにより、電話コレクション全体のデータを表示できます。

EMERGENCY_CNTCT レコード、および EMERGENCY_PHONE レコードが示すように、各レコードのキーは合成され、その親のキーとその独自の固有 ID を含んでいます。

すべてのコンポーネントが、例と同じようなわかりやすい方法でデータ・レコードを表すわけではありません。例えば、スクロール・レベルは個別のレコードまたは子レコードに常に対応するわけではありません。コンポーネントは、派生されたレコードや作業レコードを使用したり、関連する表示レコードを組み込んだり、複数のレベルで同じレコードを使用してデータを表したりすることがあります。このような状況では、コンポーネント・インターフェースの設計は例よりもさらに高度になり、処理ニーズを慎重に考慮する必要があります。

複雑なコンポーネントを使用する場合は、設計時に次の領域を考慮します。

- 『レベル』
- 49 ページの『隠しフィールド』
- 49 ページの『読み取り専用フィールド』

レベル: コンポーネントのレベルを設計する際に、次のことを考慮します。

- コンポーネント・インターフェースが、表現するコンポーネントに含まれるオンライン・ページのように動作することを確認します。コネクタが設計通りに動作するようにするには、コンポーネント・インターフェース・プロパティおよびコレクションの構造の変更か、またはユーザー定義のメソッドの追加が必要になる場合があります。
- キーは、表示される最初のレベルでのみ公開されます。ページのレベル 0 スクロールで表示されないコンポーネント・インターフェースのトップレベルのコレクションからキーを除去します。ページ定義のレベル 1 スクロールで表示されるキーを、レベル 2 コレクションに手動で追加します。

例えば、同じレコードから 3 つのキー (例えば、SETID、DEPTID、および EFFDT) を使用するページがあると仮定します。さらに、このページはスクロール・レベル 1 で EFFDT を使用して、指定された SETID および DEPTID のデータ履歴に戻すと仮定します。このページを含むコンポーネント・インターフェースを作成すると、すべてのキー・フィールドは同じレベル 0 プライマリー・レコードに存在するので、3 つのキーはすべてコレクション・レベル 0 に表示されます。このコンポーネント・インターフェースを使用してキーとして EFFDT を持つ一連の行に戻す場合、レベル 0 コレクションから EFFDT を手動で除去して、それをレベル 1 コレクションに追加する必要があります。こうすることで、コンポーネント・インターフェースがオンラインの場合と同じように動作します。

- ページ上の要素の順序に特に注意して、各コンポーネントのページ定義を調べます。各フィールドが表示されるスクロール・レベルを確認します。この情報を使用して、対応するビジネス・オブジェクト属性が正しいレベルに存在するか、または子ビジネス・オブジェクトに属しているかを判別します。

隠しフィールド: 隠しフィールドは、必ずしもコンポーネント・プロセッサにロードされるとは限りません。これらのフィールドを公開すると、アプリケーション・サービス・エラーになることがあります。ページ上で可視的に公開されないフィールドは、コネクタに公開しないことをお勧めします。

読み取り専用フィールド: コンポーネント・インターフェースの読み取り専用とマークされたフィールドは、コネクタからアクセス可能ですが、値を戻すメソッドでのみ可能です。コネクタはこれらのフィールドに値を設定できません。したがって、対応する属性のアプリケーション固有テキストの `get= FieldName` パラメーターは指定しますが、`set= FieldName` パラメーターは空にしておきます。このような場合に `set` パラメーターを指定すると、メソッドを更新または作成操作で起動した場合にエラーが発生します。

詳細については、69 ページの『属性レベルのアプリケーション固有情報』を参照してください。

図 6 に、EMER_CONTACTコンポーネントを含むコンポーネント・インターフェース (EMER_CONTACT_PROFILE) の例を示します。

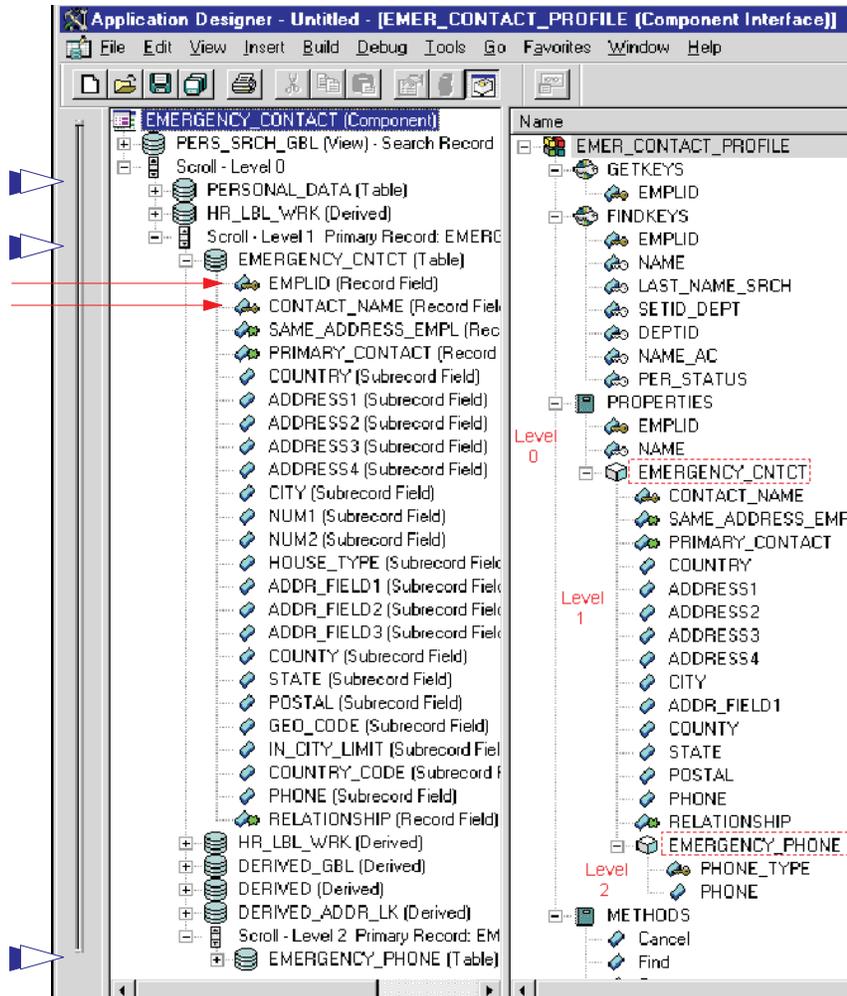


図 6. コンポーネント・インターフェースの例

左マージンにある 3 つの幅広の矢印は、コンポーネント・インターフェースのデータを格納する表 (PERSONAL_DATA、EMERGENCY_CNTCT、および EMERGENCY_PHONE) を指しています。2 つの細い矢印は、EMERGENCY_CNTCT 表のキー・フィールド (EMPLID と CONTACT_NAME) を示すキー・アイコンを指しています。

PeopleSoft では、コンポーネント・インターフェースがコンポーネントの必要な要素をサード・パーティーに公開するように設計されており、外部の統合アプリケーションに対してビジネス・プロセス・ロジック (PeopleCode、Field Edits、および PeopleSoft Security など) を透過的にします。したがって、コネクタはコンポーネント・インターフェースとデータベースの間のすべての処理をアプリケーションに依存します。

この依存関係により、検索ダイアログ処理の不足などの、ある制約が発生します。したがって、SearchInit、SearchSave、および RowSelect イベントは起動されず、これらに関連するすべての PeopleCode は実行されません。この制約は、Menu PeopleCode やポップアップ・メニューなどの GUI またはオンライン処理のみに関連するすべての PeopleCode イベントにも適用されます。

したがって、コネクタのコンポーネント・インターフェースを作成する前に、重要な動作が失われていないことと、コネクタがアクセスする前にすべての定義済みデータが配置されていることを確認します。このスキップされる動作の一部をコンポーネント・インターフェース内のユーザー定義メソッドとして、またはコンポーネント固有の PeopleCode (ビルド済みイベントなど) として実装できます。これらの対策を講じないと実行時エラーが発生することがあります。

EMER_CONTACT_PROFILE コンポーネント・インターフェースで表現されるデータをコネクタが処理できるようにするには、WebSphere Business Integration システムに PeopleSoft 固有のビジネス・オブジェクトを作成する必要があります。

ビジネス・オブジェクトの例

図 7 に、階層構造の PeopleSoft 固有のビジネス・オブジェクトの例を示します。これは、WebSphere Business Integration システムに作成できるコンポーネント・インターフェースの一例を表します。

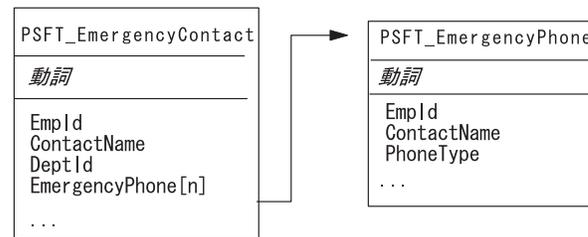


図 7. ビジネス・オブジェクトの例

PSFT_EmergencyContact ビジネス・オブジェクトには、キー・フィールド・データを表す 2 つの単純属性 (EmpId と ContactName) があります。これらの属性は、それぞれ EMERGENCY_CNTCT レコード内の EMPLID と CONTACT_NAME に対応しています。PSFT_EmergencyContact にはまた、ビジネス・オブジェクト PSFT_EmergencyPhone の配列を表す配列属性 (EmergencyPhone[n]) もあります。

PSFT_EmergencyPhone ビジネス・オブジェクトには、キー・フィールド・データを表す 3 つの単純属性 (EmpId、ContactName、および PhoneType) があります。最初の 2 つの属性はキーとして機能し、親ビジネス・オブジェクトを一意的に識別します。3 番目の属性は、同じ配列内の子とその他のビジネス・オブジェクトを一意的に区別します。

ビジネス・オブジェクトの作成

サポート・ビジネス・オブジェクトを作成するには、次の手順で行います。

1. イベントとトランザクションを表すコンポーネントを見つける。必要な場合は、作成します。
2. PeopleSoft Application Designer を使用して、ステップ 1 のコンポーネントからコンポーネント・インターフェースを作成する。
3. PeopleSoftODA または Business Object Designer を使用して、対応する PeopleSoft 固有のビジネス・オブジェクトを作成する。ビジネス・オブジェクトの詳細については、64 ページの『ビジネス・オブジェクトの属性プロパティ』と 66 ページの『ビジネス・オブジェクトのアプリケーション固有情報』を

参照してください。PeopleSoftODA の詳細については、73 ページの『第 5 章 PeopleSoftODA を使用したビジネス・オブジェクト定義の生成』を参照してください。

4. PeopleSoft 固有のビジネス・オブジェクトをコネクタのサポート・ビジネス・オブジェクトのリストに追加する。使用する統合ブローカーに固有のサブスクリプション情報については、ブローカーのインプリメンテーション・ガイドを参照してください。

PeopleSoft Application Designer を使用して、コンポーネント・インターフェースからクラス構造を作成します。コンポーネント・インターフェースに対応するビジネス・オブジェクトは、データを処理するためにこれらのクラスが必要です。データは PeopleSoft API の一部です。

生成されたクラスの作成および使用を支援するために、このセクションでは次について説明します。

- 『API の生成』
- 53 ページの『API の例』

API の生成

コンポーネント・インターフェースから API を生成するには、次を実行します。

1. アプリケーション・デザイナーでコンポーネント・インターフェースを開く。
2. 「Build」メニューから「PeopleSoft API」メニューを選択する。
3. Java オプションのみを選択して、以下のように宛先を指定する。

UNIX

```
$ProductDir/connectors/PeopleSoft/dependencies
```

UNIX の終り

Windows

```
%ProductDir%\connectors\PeopleSoft\dependencies
```

Windows の終り

4. 生成されたクラスをコンパイルして、それらが次のディレクトリに存在することを確認する。

UNIX

```
$ProductDir ディレクトリ:
```

コンポーネント・インターフェースのクラス・ファイル

```
/connectors/PeopleSoft/dependencies/PeopleSoft/Generated/CompIntfc
```

セッション固有のクラス・ファイル

```
/connectors/PeopleSoft/dependencies/PeopleSoft/Generated/PeopleSoft
```

UNIX の終り

Windows

%ProductDirS% ディレクトリー:

コンポーネント・インターフェースのクラス・ファイル

%connectors%¥PeopleSoft¥dependencies¥PeopleSoft¥Generated¥CompIntfc

セッション固有のクラス・ファイル

%connectors%¥PeopleSoft¥dependencies¥PeopleSoft¥Generated¥PeopleSoft

Windows の終り

画面のショットなどの、API 生成プロセスのさらに詳細な説明については、21 ページの『API ファイルのビルド』を参照してください。

API の例

「PeopleSoft API」メニュー（「Build メニューのサブメニュー）を使用してサンプルの EMER_CONTACT_PROFILE コンポーネント・インターフェースからクラス構造を作成する場合は、以下のものがあります。

- EmerContactProfile.class: コンポーネント・インターフェースに対応。

コネクターはこのクラス名を使用して、PeopleSoft のコンポーネント・インターフェースを検索およびインスタンス化します。クラス名は、ビジネス・オブジェクト・レベルでアプリケーション固有情報の CiName プロパティに格納されます。詳細については、67 ページの『ビジネス・オブジェクト・レベルのアプリケーション固有情報』を参照してください。

- EmerContactProfileEmergencyCntct.class: EMERGENCY_CNTCT コレクションに対応。
- EmerContactProfileEmergencyCntctEmergencyPhone.class: EMERGENCY_PHONE コレクションに対応。

コネクターが使用する、生成されたメソッドは次のとおりです。

- 『getFieldName() メソッド』
- 54 ページの 『setFieldName() メソッド』
- 54 ページの 『getCollectionName() メソッド』
- 54 ページの 『CurrentItem() メソッド』
- 55 ページの 『Item(index) メソッド』

getFieldName() メソッド

生成されたクラスにはそれぞれ `getFieldName()` メソッドがあり、このメソッドによりコネクターがコンポーネント・インターフェースの各単純フィールドのデータ値を取得し、それを対応するビジネス・オブジェクト属性にロードすることができます。

例えば、図 6 の右半分に示すように、EMER_CONTACT_PROFILE コンポーネントの FINDKEYS には 7 つのフィールドがリストされています。これらのフィールドには、EMPLID、NAME、および DEPTID が含まれています。これらのフィールドからデータを取得するために、コネクタは getEmpId()、getName()、および getDeptId() メソッドを使用します。値を取得後、コネクタはそれらの値を EmpId、ContactName、および DeptId ビジネス・オブジェクト属性にロードします。

コレクションの単純なフィールドの値を戻すために、コネクタは最初にコレクションを戻し、次にその中のフィールドを戻します。例えば、EMERGENCY_CONTACT コレクションの CONTACT_NAME フィールドと SAME_ADDRESS_EMPL フィールド内の値を取得するために、コネクタは最初に getEmergencyCntct() メソッドを実行します。次に、getContactName() と getSameAddressEmpl() メソッドを実行します。詳細については、54 ページの『getCollectionName() メソッド』を参照してください。

setFieldName() メソッド

生成されたクラスにはそれぞれ setFieldName() メソッドがあり、このメソッドによりコネクタが、対応するビジネス・オブジェクト属性の値に基づいてコンポーネント・インターフェースの各単純フィールドのデータ値を設定できます。

例えば、EmpId、ContactName、および DeptId ビジネス・オブジェクト属性から EMPLID、NAME、および DEPTID フィールドにデータをそれぞれロードするために、コネクタはそれぞれについて setEmpId()、setName()、および setDeptId() メソッドを使用します。

getCollectionName() メソッド

指定された従業員の緊急連絡先のコレクションを戻すために、コネクタは EmerContactProfile クラス内の getEmergencyCntct() メソッドを使用します。コネクタが複数行を処理する方法は、ビジネス・オブジェクト・レベルのアプリケーション固有情報の設定により異なります。

- EFFDT パラメーターの評価が true の場合、コネクタは CurrentItem() メソッドを使用して最新の有効期限を持つレコードのみを戻します。
- EFFDT パラメーターの評価が false の場合、コネクタは Item(index) メソッドを使用して、有効期限に関係なく、最初のレコードのみを戻します。例えば、コネクタは将来の日付がリストされていると、それを戻します。

EFFDT パラメーターの詳細については、67 ページの『ビジネス・オブジェクト・レベルのアプリケーション固有情報』を参照してください。

従業員の各緊急連絡先の電話のタイプと電話番号のコレクションを戻すために、コネクタは EmerContactProfileEmergencyCntct クラス内の getEmergencyPhone() メソッドを使用します。すべてのレコードを取得すると、コネクタはそれらをビジネス・オブジェクトの配列属性 EmergencyPhone[n] にロードします。

CurrentItem() メソッド

コレクション内のレコードを検索するときに、最新の有効期限を持つレコードのみを戻すには、コネクタは EmerContactProfile クラス内の CurrentItem() メソッド

ドを使用します。コネクタは、ビジネス・オブジェクト・レベルのアプリケーション固有情報の EFFDT パラメーターの評価が true の場合のみ、このメソッドを使用します。

Item(index) メソッド

コレクション内のレコードを検索するときに、指定されたレコード番号のレコードのみを戻すためには、コネクタは EmerContactProfile クラス内の Item(index) メソッドを使用します。コネクタは、ビジネス・オブジェクト・レベルのアプリケーション固有情報の EFFDT パラメーターの評価が false の場合のみ、このメソッドを使用します。デフォルトでは、このメソッドは検索した最初の行を戻します。

ビジネス・オブジェクト動詞の処理

このセクションでは、ビジネス・オブジェクト動詞の処理における以下の機能について説明します。

- 『変更後イメージとデルタ』で、用語を定義し、コネクタの変更後イメージの使用方法を説明します。
- 57 ページの『ビジネス・オブジェクト要求の動詞の処理』で、コネクタがビジネス・オブジェクトを作成、検索、更新、または削除する際のステップを説明します。
- 64 ページの『データのコミット』で、コネクタがデータを保管する方法を簡単に説明します。

変更後イメージとデルタ

変更後イメージとは、ビジネス・オブジェクトに対するすべての変更が行われた後の、ビジネス・オブジェクトの状態です。デルタとは、キー値と変更されるデータのみを格納している、更新操作で使用されたビジネス・オブジェクトです。このコネクタは、変更後イメージのみをサポートし、ビジネス・オブジェクト・デルタはサポートしません。コネクタは更新の要求ビジネス・オブジェクトを受け取ると、ビジネス・オブジェクトが更新後のデータの必要な状態を表していると仮定します。

したがって、コネクタが Update 動詞を持つ要求ビジネス・オブジェクトを受け取ると、コンポーネント・インターフェース内のビジネス・オブジェクトの現在の表現を変更して、ソース・ビジネス・オブジェクトと正確に一致するようにします。これを行うために、コネクタは単純属性値を変更して、子ビジネス・オブジェクトを追加または除去します。

コネクタが子ビジネス・オブジェクトを変更する方法の例として、仮に PSFT_EmergencyContact ビジネス・オブジェクトには、単一カーディナリティーの子を表す属性と子ビジネス・オブジェクトの配列を表す属性の 2 つの追加属性があるとします。配列の子は、それぞれ自分自身の子ビジネス・オブジェクトの配列を含むことができます。

図8に、IDが2345の従業員のPSFT_EmergencyContactの現在の状態を示します。ArrayData属性は、3つのレコード(A、B、およびC)を表します。これらのうち、2つのレコードの配列属性は、2つの追加レコードを表します。

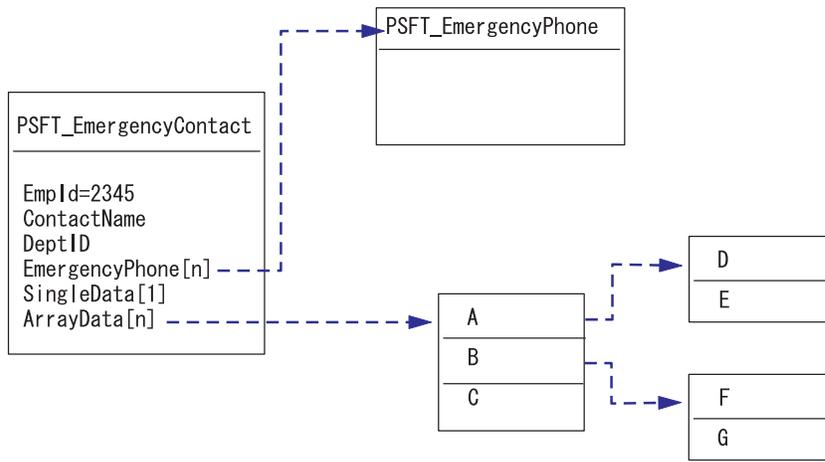


図8. 更新前のデータの状態

図9に、ビジネス・オブジェクト要求を示します。このビジネス・オブジェクトは、新規の単一カーディナリティーの子ビジネス・オブジェクトを含み、またその配列には別のビジネス・オブジェクトが含まれています。

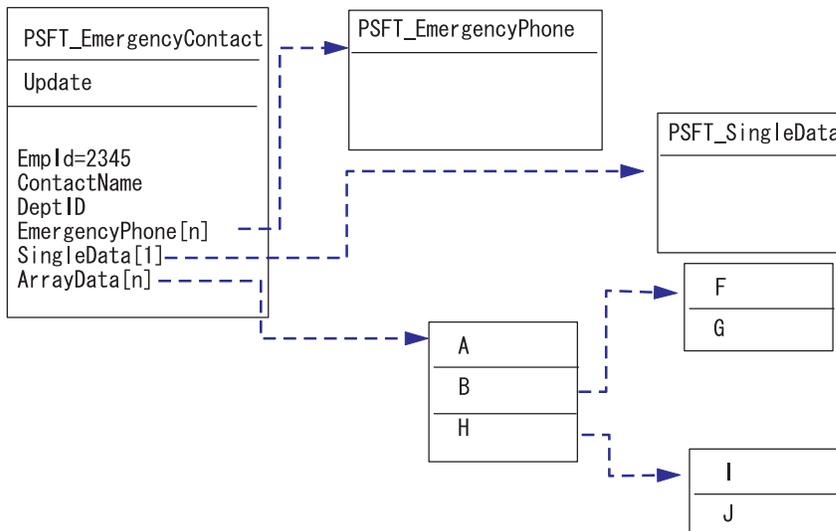


図9. 更新要求で表わされるデータ

更新を処理するために、コネクタはコンポーネント・インターフェースに次の変更を行います。

- PSFT_EmergencyContact 内の単純属性と PSFT_EmergencyPhone ビジネス・オブジェクトを更新します。

- PSFT_SingleData ビジネス・オブジェクトを作成します。
- 子ビジネス・オブジェクトの A、B、F、および G 内の単純属性を更新します。
- 子ビジネス・オブジェクトの C、D、および E を削除します。
- 子ビジネス・オブジェクトの H、I、および J を作成します。

コネクタは、受け取る各要求ビジネス・オブジェクトが変更後イメージを表していることを前提とするので、更新のためにコネクタに送られる各ビジネス・オブジェクトに有効な既存の子ビジネス・オブジェクトがすべて組み込まれていることが重要です。子ビジネス・オブジェクトの単純属性が何も変更されなくても、子ビジネス・オブジェクトがソース・ビジネス・オブジェクトに組み込まれている必要があります。

ただし、更新操作時にコネクタが、必要な子ビジネス・オブジェクトを削除しないように防止することができます。ソース・ビジネス・オブジェクトに組み込まれていない子ビジネス・オブジェクトをコネクタが保持するように指示するには、子または子の配列を表す属性に関してアプリケーション固有情報を使用します。このためには、KeepRelationship を true に設定します。詳細については、69 ページの『属性レベルのアプリケーション固有情報』を参照してください。

ビジネス・オブジェクト要求の動詞の処理

このセクションでは、コネクタが要求として受け取るビジネス・オブジェクトを作成、検索、更新、または削除する際の手順を説明します。コネクタは階層ビジネス・オブジェクトを再帰的に処理します。つまり、個々のビジネス・オブジェクトをすべて処理するまで、子ビジネス・オブジェクトそれぞれに対して同じステップを実行します。

ビジネス・オブジェクトの比較

下記に概要を説明する処理のさまざまな時点において、コネクタは 2 つのビジネス・オブジェクトを比較して、それらが同じかどうかを確認します。例えば、更新操作時に、コネクタは特定のビジネス・オブジェクトがビジネス・オブジェクトの配列内に存在するかどうかを判断します。この検査を行うために、コネクタはビジネス・オブジェクトと配列内の各ビジネス・オブジェクトを比較します。2 つのビジネス・オブジェクトが同一であるには、次の 2 つの条件を満たす必要があります。

- 比較されるビジネス・オブジェクトのタイプが同じでなければなりません。例えば、PSFT_Customer ビジネス・オブジェクトは、その属性がすべて正確に一致していても、PSFT_Contact ビジネス・オブジェクトと同一とは見なされません。
- 2 つのビジネス・オブジェクト内のすべてのキー属性には、同一の値が格納されている必要があります。両方のキー属性に CxIgnore が設定されていると、コネクタはそれらを同一と見なします。しかし、片方のビジネス・オブジェクトではキー属性に CxIgnore が設定されているが、もう一方では設定されていない場合、ビジネス・オブジェクトは同一と見なされません。

作成操作

ビジネス・オブジェクトを作成する場合、コネクタは、操作が成功したときには (操作によりビジネス・オブジェクトが変更されたかどうかに関係なく) VALCHANGE、操作が失敗したときには FAIL のいずれかの状況を戻します。

コネクタは、階層ビジネス・オブジェクトを作成する場合、次のステップで行います。

1. トップレベルのビジネス・オブジェクトのコンポーネント・インターフェースの新しいインスタンスを作成します。

- PeopleSoft の `SetFieldName(value)` メソッドを使用して、コンポーネント・インターフェース内のプロパティにビジネス・オブジェクト内の属性の値を設定します。

重要: キー属性のアプリケーション固有情報の UID パラメーターの評価が `false` の場合、ビジネス・オブジェクトを作成するビジネス・プロセスには、属性の新規の固有 ID 値が指定されている必要があります。ビジネス・オブジェクトに必要な値がない場合、コネクタはエラーをログに記録します。

キー属性のアプリケーション固有情報の UID パラメーターの評価が `true` の場合は、アプリケーションで固有 ID を生成する必要があります。この場合、属性の値または Default Value プロパティには、ストリング NEXT が設定されている必要があります。つまり、プロセス・オブジェクトを作成するビジネス・プロセスに値 NEXT が設定されていない場合は、属性の Default Value プロパティにこの値が設定されている必要があります。ビジネス・オブジェクトが Default Value プロパティを使用して、ストリング NEXT を指定する場合は、コネクタの UseDefaults プロパティの評価が `true` である必要があります。詳細については、127 ページの『PollFutureEvents』を参照してください。

属性の値に対して 固有 ID が設定されておらず、またストリング NEXT が Default Value プロパティに指定されていない場合、アプリケーションによりキーの重複エラーがログに記録されます。

- すべての Required 属性に値が設定されていることを確認し、この値が不足する場合はエラーをスローします。Required 属性の詳細については、65 ページの『Required プロパティ』を参照してください。
2. 各子ビジネス・オブジェクトと子ビジネス・オブジェクトの各配列をコンポーネント・インターフェースに再帰的に挿入します。つまり、コネクタは子ビジネス・オブジェクトと、その子に組み込まれるすべての子ビジネス・オブジェクトを作成します。

注: 単一カーディナリティーの子ビジネス・オブジェクトを表す属性のビジネス・オブジェクト定義で、子が必要 (つまり、Required プロパティの評価が `true`) と指定する場合、検索では行を戻す必要があります。行を戻さないと、コネクタはエラーを戻し、処理を停止します。ただし、子が必須でなく、属性が空の場合、コネクタは属性を無視します。

3. コネクタは、データを書き込んでコミットする `Save()` メソッドを呼び出します。
- コネクタが固有 ID を生成する場合、このメソッドの実行時に生成されません。
 - アプリケーションが固有 ID を生成する場合、コネクタはこのメソッドの実行後にアプリケーションで設定されたキー値を検索します。

注: InterChange Server (ICS) が統合ブローカーの場合、ICS はビジネス・オブジェクトを相互参照するためにこの ID が必要なので、ID を同時に送る必要があります。

- ビジネス・オブジェクトのアプリケーション固有情報に `setInteractiveMode` が `false` と定義されている場合、この時点ですべての PeopleCode 編集が行われます。すべての PeopleCode エラーは、PSMessage コレクション・キューにパブリッシュ (パブリック化) されます。
- 同じキー値を持つコンポーネント・インターフェースのインスタンスがすでに存在する場合、アプリケーションは重複エラーを戻し、コネクタが FAIL 戻りコードを送信します。

属性プロパティの詳細については、64 ページの『ビジネス・オブジェクトの属性プロパティ』を参照してください。アプリケーション固有情報の指定方法の詳細については、69 ページの『属性レベルのアプリケーション固有情報』を参照してください。

検索操作

ビジネス・オブジェクトを検索する場合、コネクタは、操作が成功したときには (操作によりビジネス・オブジェクトが変更されたかどうかに関係なく) VALCHANGE、操作が失敗したときには FAIL のいずれかの状況に戻します。

コネクタは、階層ビジネス・オブジェクトを検索する場合、次のステップで行います。

1. 統合ブローカーから受け取るトップレベルのビジネス・オブジェクトからすべての子ビジネス・オブジェクトを除去します。
2. トップレベルのビジネス・オブジェクトに対応するコンポーネント・インターフェースを検索します。

コネクタはソース・ビジネス・オブジェクト内のキー値を使用して、コンポーネント・インターフェースをインスタンス化します。検索の結果、次のいずれかのアクションが実行されます。

- コンポーネント・インターフェースのインスタンスを検出すると、コネクタは処理を継続します。
- コンポーネント・インターフェースのインスタンスを検出できず、トップレベルのビジネス・オブジェクトには、アプリケーション内に対応するコンポーネント・インターフェースがないことを示す場合、コネクタは FAIL を戻します。
- 複数のコンポーネント・インターフェースのインスタンスを検出すると、コネクタは MULTIPLE_HITS を戻します。

注: ビジネス・オブジェクトには、コンポーネント・インターフェースのどのプロパティにも対応しない属性を設定することができます。検索中に、コネクタはトップレベルのビジネス・オブジェクト内のこのような属性は変更しません。受け取った値に設定されたままです。子ビジネス・オブジェクトの場合、コネクタは検索時に、このような属性にデフォルト値を設定します。

3. ビジネス・オブジェクト配列に対応するコンポーネント・インターフェースのすべてのコレクションを再帰的に検索します。

コネクターは、それぞれの親ビジネス・オブジェクト内のキーとそれぞれの子の固有キーを使用して、コンポーネント・インターフェース・インスタンスまたはコンポーネント・インターフェース・コレクションからデータ行を選択します。戻された各行について、コネクターは次のアクションを実行します。

- a. 正しいタイプの新規の個別ビジネス・オブジェクトを作成します。
- b. 戻された行の値に基づいて、すべての現行ビジネス・オブジェクトの属性を設定します。
- c. すべての現行ビジネス・オブジェクトの子を再帰的に検索します。
- d. 現行ビジネス・オブジェクトを、そのすべての子とともに該当する親の配列に挿入します。

注: コネクターはビジネス・オブジェクトの配列に値を設定する際に、一意であることを強制しません。一意であることを確認するのは、アプリケーションで行う必要があります。アプリケーションから重複する子ビジネス・オブジェクトが戻されると、コネクターは統合ブローカーに重複する子ビジネス・オブジェクトを戻します。

4. トップレベルのビジネス・オブジェクトの単一カーディナリティーの子それぞれに関して、コレクションを再帰的に検索します。コネクターは、それぞれの親ビジネス・オブジェクト内のキーとそれぞれの子の固有キーを使用して、コンポーネント・インターフェース・インスタンスまたはコンポーネント・インターフェース・コレクションからデータ行を選択します。コネクターは、次を実行します。
 - a. ビジネス・オブジェクトの定義で子が必須であると指定されている場合、検索で行が戻される必要があります。子が必須ではなく、検索から行が戻されずにコンポーネント・インターフェース内に子が存在しないことを示すと、コネクターは親の単一カーディナリティー属性を空のままにしておきます。検索で複数の行を戻す場合、検索は失敗します。
 - b. 子ビジネス・オブジェクトにより組み込まれたすべての子をコレクションから再帰的に検索します。
 - c. ビジネス・オブジェクトを、そのすべての子とともに、親ビジネス・オブジェクトの該当する属性に挿入します。

RetrieveByContent 操作

ビジネス・オブジェクトを検索する場合、コネクターは、操作が成功したときには (操作によりビジネス・オブジェクトが変更されたかどうかに関係なく) `VALCHANGE`、操作が失敗したときには `FAIL`、操作により複数行が返されたときには `MULTIPLE_HITS` という状況に戻します。

コネクターは、階層ビジネス・オブジェクトを検索する場合、次のステップで行います。

1. 統合ブローカーから受け取るトップレベルのビジネス・オブジェクトからすべての子ビジネス・オブジェクトを除去します。
2. トップレベルのビジネス・オブジェクトに対応するコンポーネント・インターフェースを検索します。

コネクタは、ソース・ビジネス・オブジェクトの検索キーであるこれらの属性の値を使用して、コンポーネント・インターフェースをインスタンス化します。(検索キー属性に対するアプリケーション固有情報の指定の詳細については、69ページの『属性レベルのアプリケーション固有情報』を参照してください。) 検索の結果、次のいずれかのアクションが実行されます。

- コンポーネント・インターフェースのインスタンスを検出すると、コネクタは処理を継続します。
- コンポーネント・インターフェースのインスタンスを検出できず、トップレベルのビジネス・オブジェクトには、アプリケーション内に対応するコンポーネント・インターフェースがないことを示す場合、コネクタは FAIL を戻します。
- 複数のコンポーネント・インターフェースのインスタンスを検出すると、コネクタは MULTIPLE_HITS を戻します。

注: ビジネス・オブジェクトには、コンポーネント・インターフェースのどのプロパティにも対応しない属性を設定することができます。検索中に、コネクタはトップレベルのビジネス・オブジェクト内のこのような属性は変更しません。受け取った値に設定されたままです。子ビジネス・オブジェクトの場合、コネクタは検索時に、このような属性にデフォルト値を設定します。

3. ビジネス・オブジェクト配列に対応するコンポーネント・インターフェースのすべてのコレクションを再帰的に検索します。

コネクタは、それぞれの親ビジネス・オブジェクト内のキーとそれぞれの子の固有キーを使用して、コンポーネント・インターフェース・インスタンスまたはコンポーネント・インターフェース・コレクションからデータ行を選択します。戻された各行について、コネクタは次のアクションを実行します。

- a. 正しいタイプの新規の個別ビジネス・オブジェクトを作成します。
- b. 戻された行の値に基づいて、すべての現行ビジネス・オブジェクトの属性を設定します。
- c. すべての現行ビジネス・オブジェクトの子を再帰的に検索します。
- d. 現行ビジネス・オブジェクトを、そのすべての子とともに該当する親の配列に挿入します。

注: コネクタはビジネス・オブジェクトの配列に値を設定する際に、一意であることを強制しません。一意であることを確認するのは、アプリケーションで行う必要があります。アプリケーションから重複する子ビジネス・オブジェクトが戻されると、コネクタは統合ブローカーに重複する子ビジネス・オブジェクトを戻します。

4. トップレベルのビジネス・オブジェクトの単一カーディナリティーの子それぞれに関して、コレクションを再帰的に検索します。コネクタは、それぞれの親ビジネス・オブジェクト内のキーとそれぞれの子の固有キーを使用して、コンポーネント・インターフェース・インスタンスまたはコンポーネント・インターフェース・コレクションからデータ行を選択します。コネクタは、次を実行します。
 - a. ビジネス・オブジェクトの定義で子が必須であると指定されている場合、検索で行が戻される必要があります。子が必須ではなく、検索から行が戻され

ずにコンポーネント・インターフェース内に子が存在しないことを示すと、コネクターは親の単一カーディナリティー属性を空のままにしておきます。検索で複数の行を戻す場合、検索は失敗します。

- b. 子ビジネス・オブジェクトにより組み込まれたすべての子をコレクションから再帰的に検索します。
- c. ビジネス・オブジェクトを、そのすべての子とともに、親ビジネス・オブジェクトの該当する属性に挿入します。

更新操作

ビジネス・オブジェクトを更新する場合、コネクターは、操作が成功したときには (操作によりビジネス・オブジェクトが変更されたかどうかに関係なく) VALCHANGE、操作が失敗したときには FAIL のいずれかの状況に戻します。

コネクターは、階層ビジネス・オブジェクトを更新する場合、次のステップで行います。

1. ソース・ビジネス・オブジェクト内のキー値を使用して、対応するコンポーネント・インターフェースのインスタンスを検索します。検索したコンポーネント・インターフェースは、PeopleSoft アプリケーション内のデータの現在の状態を正確に表しています。
 - 検索が失敗し、トップレベルのビジネス・オブジェクトがアプリケーションに存在しないことを示すと、コネクターは BO_DOES_NOT_EXIST を戻します。
 - 検索に成功すると、コネクターは検索したコンポーネント・インターフェースとソース・ビジネス・オブジェクトを比較して、コンポーネント・インターフェース内の変換が必要な子ビジネス・オブジェクトを判別します。ただし、コネクターはソース・ビジネス・オブジェクト内の単純属性と検索したコンポーネント・インターフェース内の単純属性を比較しません。コネクターはすべての単純属性の値を更新します。
2. トップレベルのビジネス・オブジェクトのすべての単一カーディナリティーの子を再帰的に更新します。

ビジネス・オブジェクト定義で属性が子ビジネス・オブジェクトに含まれていることが必須にされている場合、ソース・ビジネス・オブジェクトと検索したコンポーネント・インターフェースの両方に子ビジネス・オブジェクトが存在する必要があります。両方に存在しない場合は、更新が失敗し、コネクターはエラーを戻します。

コネクターは、次のいずれかの方法で、単一カーディナリティーの子ビジネス・オブジェクトの更新を処理します。

- ソース・ビジネス・オブジェクトと検索したコンポーネント・インターフェースの両方に子ビジネス・オブジェクトが存在する場合、コネクターはコンポーネント・インターフェース内の子ビジネス・オブジェクトを再帰的に更新します。

注: ソース・ビジネス・オブジェクトと検索したコンポーネント・インターフェースは一致する必要があります。2 つの階層オブジェクトに同じ単一カーディナリティーを持つ子が別の順序で含まれている場合、コネクターはエラーを戻し、処理を停止します。

- ソース・ビジネス・オブジェクトには子が存在するが、検索したコンポーネント・インターフェースには存在しない場合、コネクタはコンポーネント・インターフェース内に子ビジネス・オブジェクトを再帰的に作成します。

重要: キー属性のアプリケーション固有情報の UID パラメーターの評価が `false` の場合、ビジネス・オブジェクトを作成するビジネス・プロセスには、属性の新規の固有 ID 値が指定されている必要があります。ビジネス・オブジェクトに必要な値がない場合、コネクタはエラーをログに記録します。

キー属性のアプリケーション固有情報の UID パラメーターの評価が `true` の場合は、アプリケーションで固有 ID を生成する必要があります。この場合、属性の値または `Default Value` プロパティには、`String NEXT` が設定されている必要があります。つまり、プロセス・オブジェクトを作成するビジネス・プロセスに値 `NEXT` が設定されていない場合は、属性の `Default Value` プロパティにこの値が設定されている必要があります。

属性の値に対して 固有 ID が設定されておらず、また `String NEXT` が `Default Value` プロパティに指定されていない場合、アプリケーションによりキーの重複エラーがログに記録されます。

- 検索したコンポーネント・インターフェースには子が存在するが、ソース・ビジネス・オブジェクトには存在しない場合、コネクタはコンポーネント・インターフェースから子ビジネス・オブジェクトを再帰的に削除します。ただし、親のアプリケーション固有情報の `KeepRelationship` パラメーターの評価が `true` の場合、コネクタは子ビジネス・オブジェクトを保存します。

更新操作時に子ビジネス・オブジェクトを削除する場合、コネクタは PeopleSoft の `deleteItem()` メソッドを使用して、コンポーネント・インターフェース・インスタンスから対応するコレクションを削除します。コネクタは、レベル 1 以上のコレクションのみを物理的または論理的に削除します。

アプリケーション固有情報の指定方法の詳細については、69 ページの『属性レベルのアプリケーション固有情報』を参照してください。

3. 検索したコンポーネント・インターフェースのすべての単純属性が更新されません。ただし、ソース・ビジネス・オブジェクト内の対応する属性に `CxIgnore` 値が指定されている属性は除きます。
4. 検索したコンポーネント・インターフェースのすべての配列を、次のいずれかの方法で処理します。
 - ソース・ビジネス・オブジェクトの配列と検索したコンポーネント・インターフェースの配列の両方に子が存在する場合、コネクタはコンポーネント・インターフェース内の子ビジネス・オブジェクトを再帰的に更新します。
 - ソース・ビジネス・オブジェクトの配列には子が存在するが、検索したコンポーネント・インターフェースの配列には存在しない場合、コネクタはコンポーネント・インターフェース内に子ビジネス・オブジェクトを再帰的に作成します。

- 検索したコンポーネント・インターフェースの配列には子が存在するが、ソース・ビジネス・オブジェクトの配列には存在しない場合、コネクターはコンポーネント・インターフェースから子ビジネス・オブジェクトを再帰的に削除します。

重要: ビジネス・オブジェクトを作成するビジネス・プロセスで、ソース・ビジネス・オブジェクト内の複数カーディナリティーのビジネス・オブジェクトが固有 (つまり、配列には同じビジネス・オブジェクトのコピーが複数存在しない) であることを確認する必要があります。コネクターがソース配列内で重複するビジネス・オブジェクトを受け取ると、ビジネス・オブジェクトを 2 度処理し、予測できない結果が生じることがあります。

注: コネクターは、データの整合性を確実にするために検索中はデータをロックします。

削除操作

コネクターはトップレベルのビジネス・オブジェクトは削除しません。ただし、トップレベルのビジネス・オブジェクトが Update 動詞を使用し、またソース・データを表す要求ビジネス・オブジェクトに子ビジネス・オブジェクトが存在しない場合は、子ビジネス・オブジェクトを物理的に削除します。

詳細については、62 ページの『更新操作』を参照してください。

データのコミット

コネクターは、作成または更新処理のためのビジネス・オブジェクトを受け取るたびに、コンポーネント・インターフェースにすべての変更を保管するか、またはなにも保管しません。コネクターが、データ変更の一部を保管することは決してありません。

ビジネス・オブジェクトの属性プロパティー

ビジネス・オブジェクト・アーキテクチャーで、属性に適用するさまざまなプロパティーを定義します。このセクションでは、これらのプロパティーの一部についてのコネクターの解釈方法と、それらの設定方法について説明します。

Name プロパティー

各ビジネス・オブジェクト属性には、固有な名前が必要です。

Type プロパティー

各ビジネス・オブジェクト属性には、Integer、String、または子ビジネス・オブジェクトのタイプ、などのタイプが必要です。

Cardinality プロパティー

子または子ビジネス・オブジェクトの配列を表す各ビジネス・オブジェクト属性は、この属性に、それぞれ 1 または n の値を持っています。子ビジネス・オブジェクトを表す属性はすべて、ContainedObjectVersion プロパティー (子のバージョン番号を指定) と Relationship プロパティー (Containment 値を指定) も持っています。

Key プロパティ

各ビジネス・オブジェクトの少なくとも 1 つの単純属性をキーとして指定する必要があります。このためには、このプロパティに `true` を設定します。

注: コネクタは、子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を表す属性を、キー属性として指定することはサポートしていません。

コネクタは各キー属性を使用して、コンポーネント・インターフェースのインスタンスを固有に識別または作成します。アプリケーションに固有 ID を生成させるための詳細については、57 ページの『作成操作』および 62 ページの『更新操作』を参照してください。

Required プロパティ

Required プロパティで、単純属性または単一カーディナリティーの子ビジネス・オブジェクトを表す属性に値が設定されている必要があるかどうかを指定します。

単一カーディナリティーの子ビジネス・オブジェクトを表す属性に対してこのプロパティが指定されていると、コネクタはこの属性があることにより、親のビジネス・オブジェクトに子ビジネス・オブジェクトが組み込まれていることを必要とします。

コネクタが作成要求を持つビジネス・オブジェクトを受け取る場合、コネクタは、要求された属性に有効な値が存在しないと、作成操作を失敗させます。

コネクタが検索要求を持つビジネス・オブジェクトを受け取り、ビジネス・オブジェクトに要求された属性の有効な値またはデフォルト値が存在しないと、コネクタは検索操作を失敗させます。

コネクタは、子ビジネス・オブジェクトの配列を表す属性に対して、このプロパティを使用しません。

Max length プロパティ

属性がストリング・タイプの場合、このプロパティで属性値に許容される最大長を指定します。

AppSpecificInfo

このプロパティの詳細については、69 ページの『属性レベルのアプリケーション固有情報』を参照してください。

Default value プロパティ

このプロパティで、属性に値が設定されていない場合にコネクタが単純なフィールドに設定するデフォルト値を指定します。コネクタは、子ビジネス・オブジェクトを表す属性に対して、このプロパティを評価しません。作成操作では、コネクタは、このプロパティの値を使用します。

コネクタは、次が `true` の場合、このプロパティの値をアプリケーションに送って、ID の生成に使用します。

- アプリケーション固有情報で UID パラメーターに true が指定されているキー属性に対して、Default Value プロパティが指定されている。
- Default Value プロパティの値が、ストリング NEXT である。
- コネクターの UseDefaults プロパティの評価が true である。

アプリケーションに固有 ID を生成させるための詳細については、57 ページの『作成操作』および 62 ページの『更新操作』を参照してください。

特殊属性値

ビジネス・オブジェクトの単純属性は、特殊な値 (CxIgnore) を持つことができます。コネクターは、要求ビジネス・オブジェクトを受け取る場合、CxIgnore の値を持つ属性をすべて無視します。これらの属性がコネクターで参照できないかのように処理されます。

コネクターは、ヌル値が指定されたフィールドを持つデータをコンポーネント・インターフェースから検索すると、デフォルトでは、これに対応する属性の値として CxIgnore を設定します。

コネクターは、ビジネス・オブジェクトを作成するために少なくとも 1 つのキー属性を必要とするので、ビジネス・オブジェクトを作成するビジネス・プロセスで、コネクターに渡されるビジネス・オブジェクトには、CxIgnore に設定されていないキーが少なくとも 1 つあることを確認する必要があります。この要件の唯一の例外は、キーがコネクターで生成されるビジネス・オブジェクトです。

ビジネス・オブジェクトのアプリケーション固有情報

ビジネス・オブジェクト定義内のアプリケーション固有情報で、ビジネス・オブジェクトの処理方法についてのアプリケーション依存の指示がコネクターに与られます。コネクターは、ビジネス・オブジェクトの属性または動詞、あるいはビジネス・オブジェクト自体からのアプリケーション固有情報を構文解析して、作成、更新、検索、または削除操作の照会を生成します。

コネクターは、ビジネス・オブジェクトのアプリケーション固有情報の一部をキャッシュに格納し、この情報を使用してすべての動詞の照会をビルドします。

アプリケーション固有のビジネス・オブジェクトを拡張または変更する場合、ビジネス・オブジェクト定義内のアプリケーション固有情報が、コネクターが予想する構文に一致することを確認する必要があります。

このセクションでは、コネクターでサポートされるビジネス・オブジェクトのアプリケーション固有情報のフォーマットについて説明します。

表 4 で、ビジネス・オブジェクトのアプリケーション固有情報で利用できる機能の概要を説明します。

表4. サポートされているビジネス・オブジェクトのアプリケーション固有情報の概要

アプリケーション固有情報の有効範囲	
有効範囲	機能性
ビジネス・オブジェクト全体	次を指定します。 <ul style="list-style-type: none"> • 対応するコンポーネント・インターフェースの名前。 • オンライン Field PeopleCode 編集のトリガーを、即時、またはバッチのいずれで行うか。 • コンポーネント・インターフェースに関連するすべての行、または一番最近のインスタンスのみのいずれを検索するか。
単純属性	次を指定します。 <ul style="list-style-type: none"> • アプリケーションで固有 ID 値を生成する、またはコネクタで 固有 ID 値を用意するのいずれか。 • データを検索するキーとして属性を使用するかどうか。
ビジネス・オブジェクト用動詞	コネクタは、動詞に基づいて機能を指定しません。

次のセクションで、この機能についてさらに詳細に説明します。

ビジネス・オブジェクト・レベルのアプリケーション固有情報

ビジネス・オブジェクト・レベルのアプリケーション固有情報により、次のことが可能です。

- 対応するコンポーネント・インターフェースの名前を指定する。
- オンライン・フィールド PeopleCode を、更新後即時に起動するか、または save() メソッド実行時に起動するかを定義する。
- 有効期限に関係なく、コンポーネント・インターフェースからすべての行を検索するかどうかを指示する。

ビジネス・オブジェクト・レベルでは、アプリケーション固有情報のフォーマットはコロン (;) 区切り文字で区切られた 4 つのパラメーターで構成されます。フォーマットは、次のとおりです。

```
cIName=<ComponentInterface>
:EFFDT=[true|false]:setInteractiveMode=[true|false]
:GetHistoryItems
=[true|false]:setEditHistoryItems=[true|false]
:GetDummyRows=[true|false]:InsAtOldestEffDtPos=[true]
:InsAtCurrentEffDtPos=[true]
:processAttributesInOrder=[true|false]
```

表5 で、これらのパラメーターについて説明します。

表5. ビジネス・オブジェクト・レベルの AppSpecificInfo

AppSpecificinfo パラメーター	説明
cIName	PeopleSoft アプリケーションに定義されたコンポーネント・インターフェースの名前を指定します。

表5. ビジネス・オブジェクト・レベルの *AppSpecificInfo* (続き)

AppSpecificInfo パラメーター	説明
EFFDT (isEffectiveDated)	<p>ビジネス・オブジェクト (または子ビジネス・オブジェクト) が有効期限を使用するかどうかを指定します。</p> <ul style="list-style-type: none"> EFFDT パラメーターの評価が <code>true</code> の場合、コネクタは <code>CurrentItem()</code> メソッドを使用して最新の有効期限を持つレコードのみを戻します。 EFFDT パラメーターの評価が <code>false</code> の場合、コネクタは <code>Item(index)</code> メソッドを使用して、有効期限に関係なく、最初のレコードのみを戻します。例えば、コネクタは将来の日付がリストされていると、それを戻します。
setInteractiveMode	<p>デフォルト値は <code>true</code> です。 コネクタが変更をアプリケーション・サーバーに送信する時期を判別します。このプロパティは、通常、パフォーマンスを向上するために使用されます。</p> <ul style="list-style-type: none"> <code>true</code> に設定されていると、コネクタは、プロパティまたは属性の値が変更されるたびに処理します。つまり、コネクタが各属性について <code>setPropertyName()</code> を呼び出した後で、即時にすべてのオンライン <code>Field PeopleCode</code> を起動します。 <code>false</code> に設定されていると、PeopleSoft アプリケーションによりバッチ処理され、コネクタがコンポーネント・インターフェースで <code>Save()</code> を呼び出すときに、変更がアプリケーション・サーバーに送信されます。
GetHistoryItems	<p>デフォルト値は <code>true</code> です。 コネクタが検索するデータ量を決定します。</p> <ul style="list-style-type: none"> <code>true</code> に設定されていると、コネクタは、有効期限に関係なく、対応するコンポーネント・インターフェースのすべてのデータ行を検索します。 <code>false</code> に設定されていると、コネクタは現在のデータ (有効な行) のみを検索します。
GetDummyRows	<p>デフォルト値は <code>true</code> です。 PeopleTools 8.4 以上のバージョンの <code>GetDummyRows</code> プロパティをサポートします。ただし、これより低いバージョンの PeopleTools においても、これを <code>true</code> に設定してください。このパラメーターをサポートしない、バージョン 8.4 より低いバージョンの PeopleTools を使用している場合は、アダプターによって警告がログに記録される場合があります。これらの警告は無視できます。このプロパティの詳細については、PeopleSoft の資料を参照してください。</p>

表 5. ビジネス・オブジェクト・レベルの *AppSpecificInfo* (続き)

AppSpecificInfo パラメーター	説明
InsAtOldestEffDtPos	このパラメーターは、 <code>getEffectiveItemNum()</code> 呼び出しによって有効期限の挿入位置に -1 が返される PeopleTools の以前のバージョンに制限を加えるために設定します。 InsAtOldestEffDtPos が設定されていると、コネクターはこの行を索引の最上部、つまり有効期限が最も古い日付の位置に挿入します。このパラメーターを設定する場合は、InsAtCurrentEffDtPos を設定しないでください。
InsAtCurrentEffDtPos	このパラメーターは、 <code>getEffectiveItemNum()</code> 呼び出しによって有効期限の挿入位置に -1 が返される PeopleTools の以前のバージョンに制限を加えるために設定します。 InsAtCurrentEffDtPos が設定されていると、コネクターはこの行を索引の最下部ゼロ、つまり有効期限が最も新しい日付の位置に挿入します。このパラメーターを設定する場合は、InsAtOldestEffDtPos を設定しないでください。
processAttributesInOrder	アダプターがビジネス・オブジェクト定義にリストされる順序で属性を処理できるようにします。
SetEditHistoryItems	使用可能な編集およびデータ履歴の保管。有効期限フィールドでのみ使用されます。

setInteractiveMode と GetHistoryItems プロパティの詳細については、PeopleSoft の資料を参照してください。

例えば、PSFT_EmergencyContact ビジネス・オブジェクトには、ビジネス・オブジェクトのアプリケーション固有情報として次の値が指定されている場合があります。

```
cIName=EMER_CONTACT_PROFILE:setInteractiveMode=false:
GetHistoryItems=true:isEffectiveDated=false
```

属性レベルのアプリケーション固有情報

属性レベルのアプリケーション固有情報で、属性単位のレベルでのコネクターの動作を指定します。アプリケーション固有情報のフォーマットは、それぞれにパラメーター名と値を持つ、5 セットの名前値パラメーターです。各パラメーター・セットは、コロン (:) 区切り文字で区切られます。垂直バー (|) は、オプション・セットのメンバーを区切ります。フォーマットは、次のとおりです。

```
get=getFieldName:set=setFieldName:UID=[true|false]:GetKey=[true|false]
:KeepRelationship
=[true|false]:findKey=[true|false]:bigDec=true:
EFFDT=[true|false]
:EFFDTSEQ=[true|false]
```

重要: コネクターがアプリケーション固有情報を評価する場合、大文字と小文字を区別します。

例えば、データを検索する際にキー・フィールドに加えて使用される Read-Only 単純属性の場合、次のフォーマットで指定できます。

```
get=getBusinessUnit:GetKey=true
```

子を保存する必要がある複数カーディナリティー属性の場合、子が更新ビジネス・オブジェクト要求に組み込まれていなくても、次のフォーマットで指定できます。

```
get=getEmergencyPhone:set=setEmergencyPhone
:KeepRelationship=true
```

表 6 で、各名前値パラメーターを説明します。

表 6. アプリケーション固有情報の属性の名前値のパラメーター

パラメーター	説明
get=getFieldName	このパラメーターには、コンポーネント・インターフェース・フィールドか、または現在の属性のコンポーネント・インターフェース・コレクションを検索する場合に使用するメソッドを指定します。
set=setFieldName	このパラメーターには、現在の属性の値に基づいて、コンポーネント・インターフェース・フィールドかまたはコンポーネント・インターフェース・コレクションに値を設定する場合に使用するメソッドを指定します。
UID=[true false]	<p>このパラメーターが <code>false</code> に設定されていると、属性の固有な値をビジネス・オブジェクトを作成するビジネス・プロセスで指定する必要があります。属性に値が指定されていないと、コネクタはエラー・メッセージを送信します。</p> <p>このパラメーターが <code>true</code> に設定されていると、アプリケーションで (自動番号付けを使用して) 固有 ID を生成する必要があります。ID 生成で、ストリング NEXT を指定するために属性値が必要です。ビジネス・オブジェクトを作成するビジネス・プロセスで、その値に NEXT を設定するか、または属性の Default Value プロパティーでこの値を指定する必要があります。ビジネス・オブジェクトが Default Value プロパティーを使用して、ストリング NEXT を指定する場合は、コネクタの UseDefaults プロパティーの評価が <code>true</code> である必要があります。詳細については、127 ページの『PollFutureEvents』を参照してください。</p> <p>キー値を <code>varchar</code> 以外の値に設定した場合、アプリケーションで ID を生成させるには、値をブランクにして <code>UID=True</code> に設定してください。キー値が <code>varchar</code> データ・タイプである場合のみ、値を NEXT に設定してください。</p> <p>アプリケーションで固有 ID を生成する場合、コネクタは、保管操作が完了してデータがコミットされた後で、アプリケーションで生成された ID を検索します。</p> <p>ICS が統合ブローカーの場合、コネクタは検索した ID を使用してビジネス・オブジェクトを相互参照します。</p> <p>属性が、アプリケーションで固有 ID を生成することを必要としていない場合、この値を <code>false</code> に設定するか、またはこのパラメーターをアプリケーション固有情報に組み込まないようにします。</p>

表 6. アプリケーション固有情報の属性の名前値のパラメーター (続き)

パラメーター	説明
GetKey=[true false]	<p>このパラメーターが <code>true</code> に設定されて、属性の <code>Key</code> プロパティと <code>Required</code> プロパティの評価が <code>true</code> の場合、コネクタは検索または更新の検索操作時に、キーの一部としてこの属性を組み込みます。</p>
KeepRelationship=[true false]	<p>コネクタはこのパラメーターを使用して、区別が必要な場合に検索キーと作成キーを区別します。作成操作を実行する場合、コネクタはキーとして、その <code>Key</code> プロパティおよび <code>Required</code> プロパティの評価が <code>true</code> の属性のみを指定します。作成キー・フィールドだけでは必ずしも必要なデータを固有に検索しないので、コネクタは検索操作を実行する際に、その <code>GetKey</code> パラメーターの評価が <code>true</code> のすべての属性をキーに追加します。このパラメーターは、子ビジネス・オブジェクトの配列を表す属性にのみ使用され、更新操作時にソース・ビジネス・オブジェクトで表されない既存の子ビジネス・オブジェクトをコネクタが削除するかどうかを指定します。</p> <ul style="list-style-type: none"> • 削除を防止するには、<code>true</code> に設定します。 • 削除を許可するには、<code>false</code> に設定します。 <p>例えば、既存の電話番号が既存の連絡先と関連していると仮定します。さらに、コネクタは単一の子ビジネス・オブジェクトを含む <code>PSFT_EmergencyContact</code> ビジネス・オブジェクトを更新する要求を受け取ると仮定します。子は、緊急電話番号と連絡先を関連付けます。<code>EmergencyPhone[n]</code> 属性の <code>KeepRelationship</code> の評価が <code>true</code> の場合、コネクタは、既存の関連を削除しないで、新規の関連を追加して連絡先を更新します。</p> <p>ただし、<code>KeepRelationship</code> の評価が <code>false</code> の場合、コネクタはソース・ビジネス・オブジェクトに組み込まれていない既存のすべての子データを削除します。このような場合、連絡先は新規の電話番号にのみ関連付けされます。</p>
findKey=[true false]	<p>このパラメーターが <code>true</code> に設定されている場合、コネクタは内容による検索の操作時に、この属性をキーとして組み込みます。このパラメーターは、コンポーネント・インターフェースの検索キー属性に対して設定してください。</p>
bigDec=[true false]	<p>このパラメーターによって、<code>BigDecimal</code> タイプ属性が識別できます。コンポーネント・インターフェースの <code>BigDecimal</code> 属性のこのパラメーターには <code>true</code> を設定してください。</p>
EFFDT=[true false]	<p>このパラメーターによって、コンポーネント・インターフェースの有効期限属性が識別できます。コネクタは、ビジネス・オブジェクト・レベルでアプリケーション固有情報の <code>EFFDT</code> が <code>true</code> に設定されている場合にのみこのパラメーターを使用します。</p>
EFFDTSEQ=[true false]	<p>このパラメーターによって、<code>getEffectiveItemNum</code> から返される挿入位置に基づいた有効期限属性の順序が識別できます。コネクタは、ビジネス・オブジェクト定義に有効期限属性が含まれている場合にのみこのパラメーターを使用します。</p>

第 5 章 PeopleSoftODA を使用したビジネス・オブジェクト定義の生成

この章では、IBM WebSphere Business Integration Adapter for PeopleSoft のビジネス・オブジェクト定義を生成する PeopleSoftODA (ODA (object discovery agent)) について説明します。コネクタは PeopleSoft コンポーネント・インターフェースとそれに関連するコレクションに基づくオブジェクトを使用するので、PeopleSoftODA はコンポーネント・インターフェース Java API を使用して、PeopleSoft データ・ソースに固有なビジネス・オブジェクト要件を検索します。

この章は、以下のセクションから構成されています。

- 『インストールと使用法』
- 77 ページの『Business Object Designer での PeopleSoftODA の使用』
- 85 ページの『生成された定義の内容』
- 88 ページの『ビジネス・オブジェクト定義ファイルのサンプル』
- 95 ページの『ビジネス・オブジェクト定義への情報の追加』

インストールと使用法

このセクションでは、以下について説明します。

- 『PeopleSoftODA のインストール』
- 74 ページの『PeopleSoftODA を使用する前に』
- 75 ページの『PeopleSoftODA の起動』
- 75 ページの『PeopleSoftODA の複数インスタンスの実行』
- 76 ページの『エラーおよびトレース・メッセージ・ファイルの処理』

PeopleSoftODA のインストール

PeopleSoftODA をインストールするには、IBM WebSphere Business Integration Adapters 用のインストーラーを使用します。「MQ Integrator Broker 用インプリメンテーション・ガイド」の説明に従ってください。InterChange System (ICS) の場合は「システム・インストール・ガイド (UNIX 版)」または「システム・インストール・ガイド (Windows 版)」の説明に従ってください。インストールが完了すると、次のファイルがシステムの製品ディレクトリーにインストールされます。

- ODA¥PeopleSoftODA.jar
- ODA¥PeopleSoftODAAgent.txt
- ODA¥messages¥PeopleSoftODAAgent_II_TT.txt ファイル (言語 (_II) および国/地域 (_TT) に固有のメッセージ・ファイル)
- ODA¥start_PeopleSoftODA.bat (Windows のみ)
- ODA/PeopleSoft/start_PeopleSoftODA.sh (UNIX のみ)

注: 特に明記されている場合を除き、この資料では、ディレクトリー・パスに円記号 (¥) を使用します。UNIX システムの場合には、円記号はスラッシュ (/) に

置き換えてください。すべての製品のパス名は、ご使用のシステムで製品がインストールされているディレクトリーを基準とした相対パス名です。

PeopleSoftODA を使用する前に

PeopleSoftODA を実行する前に、システムに必要なファイルが存在し、また ODA を実行するスクリプトまたはバッチ・ファイルに変数が正しく設定されていることを確認します。この章では、コネクターのインストールの指示にすでに従っていることを前提としています。コネクターのインストールの一部として、次のセクションでの指示に従ってください。

- 14 ページの『必要なソフトウェア』: psjoa.jar のダウンロードについての説明
- 18 ページの『コネクター用のアプリケーションを使用可能にする』: PSFTCI.jar の作成についての説明

したがって、製品ディレクトリー下の connectors¥PeopleSoft¥dependencies ディレクトリーで、次のファイルを検出できます。

- psjoa.jar: PS_HOME ディレクトリーの ¥web¥PSJOA からダウンロード。
PeopleSoftODA はこのファイルを使用して、コンポーネント・インターフェースとその情報を、PeopleSoft アプリケーション・サーバーの Jolt 部を使用して同期して送信します。
- PSFTCI.jar: Application Designer のコンポーネント・インターフェース定義から作成。PeopleSoft でコンポーネント・インターフェース API ファイルを生成後、それをコンパイルする必要があります。PeopleSoftODA は、このファイルを使用してビジネス・オブジェクト定義を生成します。詳細については、52 ページの『API の生成』を参照してください。

重要: 整合性を確実に保つために、PeopleSoftODA を実行する前にすべてのコンポーネント・インターフェースを再生成して再コンパイルすることをお勧めします。コンポーネント・インターフェースが JAR ファイル内に存在しない場合、または上記の JAR ファイルが正しいディレクトリーに存在しない場合は、始動スクリプトまたはバッチ・ファイルを修正して検出できるようにします。

シェルまたはバッチ・ファイルを編集のために開いて、表 7 で説明されている値が正しいことを確認します。

表 7. シェルおよびバッチ・ファイルの構成変数

変数	説明	例
set AGENTNAME	ODA の名前	set AGENTNAME = PeopleSoftODA
set AGENT	ODA の JAR ファイル の名前	UNIX: set AGENT = \${ProductDirS}/ODA/PeopleSoft/PeopleSoftODA.jar WINDOWS: set AGENT = %ProductDirS%¥ODA¥PeopleSoft¥PeopleSoftODA.jar

注: PeopleSoftODA を CORBA オブジェクトとして登録するか、または OAD (Object Activation Daemon) に登録する場合、Object Discovery Agent 登録ウィザードを使用して PeopleSoft ドライバーのクラス・パスを変更できます。ODA の登録については、「システム・インストール・ガイド (UNIX 版)」または「システム・インストール・ガイド (Windows 版)」を参照してください。

PeopleSoft ドライバーをインストールし、シェルまたはバッチ・ファイルに構成値を設定後、次を実行してビジネス・オブジェクトを生成する必要があります。

1. ODA を起動します。
2. Business Object Designer を起動します。
3. Business Object Designer の 6 つのステップの処理を実行して、ODA を構成し、実行します。

このステップについては、以下のセクションで詳しく説明します。

PeopleSoftODA の起動

適切なスクリプトを使用して PeopleSoftODA を起動します。

```
----- UNIX -----  
start_PeopleSoftODA.sh
```

```
----- UNIX の終了 -----
```

```
----- Windows -----  
start_PeopleSoftODA.bat
```

```
----- Windows の終了 -----
```

PeopleSoftODA の構成と実行には、Business Object Designer を使用します。Business Object Designer は、各スクリプト・ファイルまたはバッチ・ファイルの AGENTNAME 変数に指定された名前により各 ODA を見つけます。このコネクタに対するデフォルト ODA の名前は、PeopleSoftODA です。インストール中に ODA を Object Activation Daemon に登録すると、ウィザードが AGENTNAME 値の前に自動的にホスト名を追加することにより、固有なエージェント名を作成します。

PeopleSoftODA の複数インスタンスの実行

複数インスタンスの ODA を実行する場合は、ODA の名前を変更することをお勧めします。固有の名前の PeopleSoftODA インスタンスを追加するには、次の手順で行います。

- インスタンスごとに別個のスクリプトまたはバッチ・ファイルを作成する。
- 各スクリプトまたはバッチ・ファイルの AGENTNAME 変数に固有の名前を指定する。

別のマシンで ODA インスタンスを実行する場合は、各名前の接頭部にホスト・マシン名を付加することをお勧めします。ODA を OAD に登録している場合、ORB 検出機能 (osfind) を使用して、ネットワーク上の既存の CORBA オブジェクト名を見つけることができます。

78 ページの図 10 に、ODA を選択して実行する Business Object Designer のウィンドウを示します。

注: 各 ODA インスタンスの接続プロパティが同一である必要があります。追加される ODA インスタンスは、最初に接続するアプリケーション・サーバー・インスタンスに接続する必要があります。

エラーおよびトレース・メッセージ・ファイルの処理

エラーおよびトレース・メッセージ・ファイル (デフォルトは PeopleSoftODAAgent.txt) は製品ディレクトリ下の ¥ODA¥messages¥ に配置されます。これらのファイルには、次の命名規則が使用されます。

AgentNameAgent.txt

ODA スクリプト・ファイルまたはバッチ・ファイルの複数のインスタンスを作成し、各インスタンスに対応する ODA に固有の名前を指定した場合には、各 ODA インスタンスに対応するメッセージ・ファイルを持つことができます。異なる名前の付いた ODA インスタンスが複数存在しても、メッセージ・ファイルは共通にすることも可能です。有効なメッセージ・ファイルを指定する方法は 2 つあります。

- ODA の名前を変更し、それに対応するメッセージ・ファイルを作成しない場合には、ODA 構成の一部として、Business Object Designer でメッセージ・ファイルの名前を変更する必要があります。Business Object Designer はメッセージ・ファイルの名前を指定しますが、実際にファイルを作成するわけではありません。ODA 構成の一部として表示されたファイルが存在しない場合には、既存のファイルを指すように値を変更してください。
- 特定の ODA に対応する既存のメッセージ・ファイルをコピーし、必要に応じて変更することもできます。Business Object Designer は、各ファイルが命名規則に従って命名されることを前提としています。例えば、AGENTNAME 変数が PeopleSoftODA1 を指定する場合、Business Object Designer は、対応するメッセージ・ファイルの名前が PeopleSoftODA1Agent.txt であると想定します。したがって、Business Object Designer が確認のため ODA 構成の一部としてファイル名を提供するとき、このファイル名は ODA 名に基づいています。デフォルトのメッセージ・ファイルが正しく命名されていることを確認し、必要ならば訂正してください。

重要: ODA の構成時にメッセージ・ファイルの名前を正しく指定できなかった場合には、ODA はメッセージなしに稼働します。メッセージ・ファイル名の指定方法の詳細については、78 ページの『初期化プロパティの構成』を参照してください。

構成プロセスの間に、以下の項目を指定します。

- PeopleSoftODA がエラー情報およびトレース情報を書き込むファイルの名前。
- 0 から 5 までの範囲のトレース・レベル。

表 8 で、これらの値を説明します。

表 8. トレース・レベル

トレース・レベル	説明
0	すべてのエラーを記録します。
1	メソッドのすべての開始メッセージおよび終了メッセージをトレースします。
2	ODA のプロパティとそれらの値をトレースします。
3	すべてのビジネス・オブジェクトの名前をトレースします。
4	作成されたすべてのスレッドの詳細をトレースします。
5	<ul style="list-style-type: none"> • そのすべてのプロパティに関する ODA 初期設定値を示します。 • PeopleSoftODA が作成した各スレッドの詳細な状態をトレースします。 • ビジネス・オブジェクト定義ダンプをトレースします。

これらの値の構成方法については、78 ページの『初期化プロパティの構成』を参照してください。

Business Object Designer での PeopleSoftODA の使用

このセクションでは、Business Object Designer で PeopleSoftODA を使用して、ビジネス・オブジェクト定義を生成する方法について説明します。Business Object Designer の起動方法については、「ビジネス・オブジェクト開発ガイド」を参照してください。

ODA の起動後、Business Object Designer を起動させ、ODA を構成して実行します。Business Object Designer で ODA を使用してビジネス・オブジェクト定義を生成する手順は、6 つのステップから構成されます。Business Object Designer は、これらのステップを順にガイドしていくウィザードを提供します。

ODA の起動後、このウィザードを起動するには、次の手順を実行します。

1. Business Object Designer を開きます。
2. 「ファイル」メニューから、「ODA を使用して新規作成...」サブメニューを選択します。

Business Object Designer に、ウィザードの最初のウィンドウ（「エージェントの選択」という名前）が表示されます。78 ページの図 10 に、このウィンドウを示します。

ODA を選択、構成、および実行するには、以下のステップを実行してください。

1. 78 ページの『ODA の選択』
2. 78 ページの『初期化プロパティの構成』
3. 80 ページの『ノードの展開とコンポーネント・インターフェースおよびコレクションの選択』
4. 81 ページの『オブジェクトの選択の確認』
5. 82 ページの『定義の生成』。必要であれば 83 ページの『追加情報の入力』
6. 84 ページの『定義の保管』

ODA の選択

図 10 に、Business Object Designer の 6 ステップのウィザードの最初のダイアログ・ボックスを示します。このウィンドウで、実行する ODA を選択します。

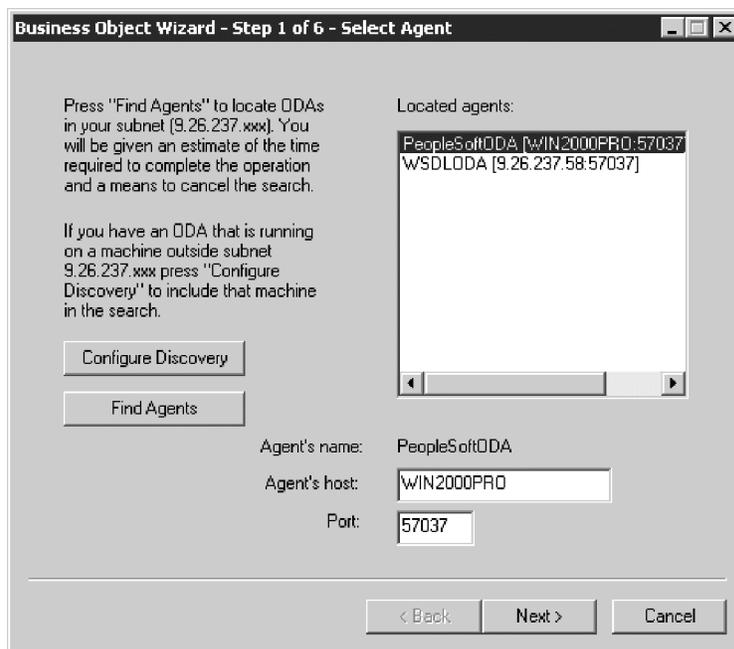


図 10. ODA の選択

ODA を選択するには、以下の手順を行います。

1. 「エージェントの検索」ボタンをクリックすることにより、登録済みまたは現在実行中の ODA のすべてを「検索されたエージェント」フィールドに表示します。

注: Business Object Designer で必要な ODA を検出できない場合は、ODA のセットアップを調べてください。

2. 表示リストから、目的の ODA を選択します。

Business Object Designer の「エージェント名」フィールドに、選択した ODA が表示されます。

初期化プロパティの構成

Business Object Designer は、PeopleSoftODA と初めて通信する際に、一連の初期化プロパティの入力を要求します。その画面を図 11 に示します。これらのプロパティは、PeopleSoftODA を使用するたびに入力せずに済むように、名前を付けたプロファイルに保存できます。ODA プロファイルの指定方法については、「ビジネス・オブジェクト開発ガイド」を参照してください。

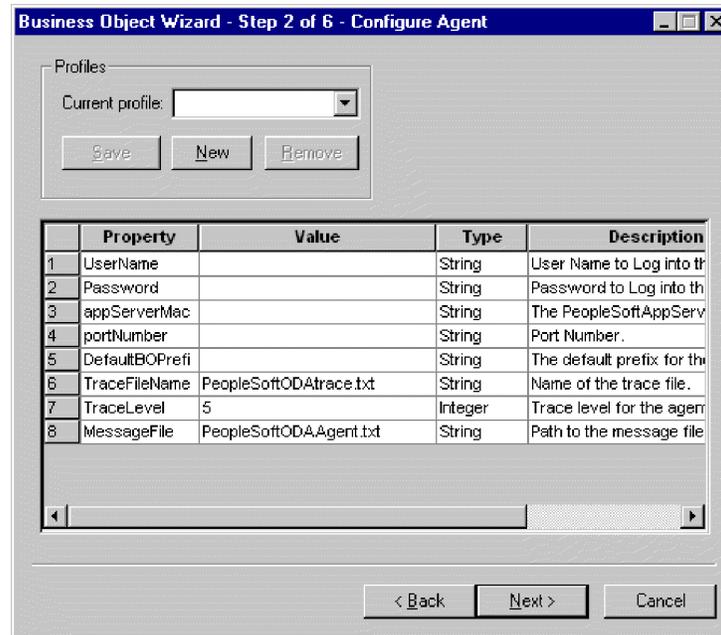


図 11. エージェント初期化プロパティの構成

PeopleSoftODA プロパティの構成を表 9 に示します。

表 9. PeopleSoftODA プロパティ

行番号	プロパティ名	プロパティ・タイプ	説明
1	UserName	String	アプリケーション・サーバーを使用して PeopleSoft アプリケーションに接続する権限を持つユーザー名。
2	Password	String	PeopleSoft アプリケーションに接続する権限を持つユーザーのパスワード。
3	AppServerMachineName	String	PeopleSoft サーバーが実行するマシンの名前または IP アドレス。
4	PortNumber	String	アプリケーション・サーバーに接続するために使用する Jolt ポート。デフォルト値は 9000 です。注: このポートは Tuxedo ポートとは異なります。
5	DefaultBOPrefix	String	ビジネス・オブジェクトの名前を固有にするために、名前の前に付加するテキスト。プレフィックスはアンダースコア () で始まることはできません。必要であれば、Business Object Designer によりビジネス・オブジェクト・プロパティのプロンプトが表示されたときに、この値を変更できます。詳細については、83 ページの『追加情報の入力』を参照してください。

表 9. PeopleSoftODA プロパティ (続き)

行番号	プロパティ名	プロパティ・タイプ	説明
6	TraceFileName	String	PeopleSoftODA によりトレース情報が書き込まれるファイル。ファイルが存在しない場合、PeopleSoftODA はファイルを %ODA%\PeopleSoft ディレクトリーに作成します。ファイルがすでに存在している場合、PeopleSoftODA はトレース情報をファイルの後に追加します。PeopleSoftODA は次の命名規則に従ってファイルに名前を付けます。例えば、エージェントの名前が PeopleSoftODA の場合、PeopleSoftODAttrace.txt という名前のトレース・ファイルを生成します。このプロパティを使用して、このファイルとは異なる名前を指定します。
7	TraceLevel	Integer	PeopleSoftODA に対して有効なトレースのレベルです。詳細については、76 ページの『エラーおよびトレース・メッセージ・ファイルの処理』を参照してください。
8	MessageFile	String	エラー/メッセージ・ファイルの名前。PeopleSoftODA は次の命名規則に従ってファイル名を表示します。例えば、エージェントの名前が PeopleSoftODA の場合、メッセージ・ファイル・プロパティの値は PeopleSoftODAAgent.txt と表示されます。 重要: エラーおよびメッセージ・ファイルは %ODA%\messages ディレクトリーに配置する必要があります。このプロパティを使用して、既存のファイルの確認や指定をします。

重要: Business Object Designer で表示されているデフォルト値が存在しないファイルを指している場合には、メッセージ・ファイルの名前を訂正します。このダイアログ・ボックスから移動したときに、名前が不正確であった場合に、Business Object Designer は、ODA の起動元となったウィンドウにエラー・メッセージを表示します。このメッセージは、Business Object Designer ではポップアップしません。有効なメッセージ・ファイルの指定に失敗すると、ODA はメッセージなしに稼働します。

ノードの展開とコンポーネント・インターフェースおよびコレクションの選択

Business Object Designer は以前のステップで構成されたプロパティを使用して、ツールを指定された PeopleSoft アプリケーションに接続する接続ストリングを作成します。接続後、Business Object Designer により、ノードが PeopleSoft アプリケーションで定義されているすべてのコンポーネント・インターフェースを表すツリーが表示されます。

ノードをクリックして、次レベルのコンポーネント・インターフェースのコレクションを表示します。コンポーネント・インターフェースを展開して、全体の階層表現を表示できます。各コレクションについて、PeopleSoftODA は子ビジネス・オブジェクト定義を作成します。

図 12 に、一部のコンポーネント・インターフェースが展開された、このダイアログ・ボックスを示します。

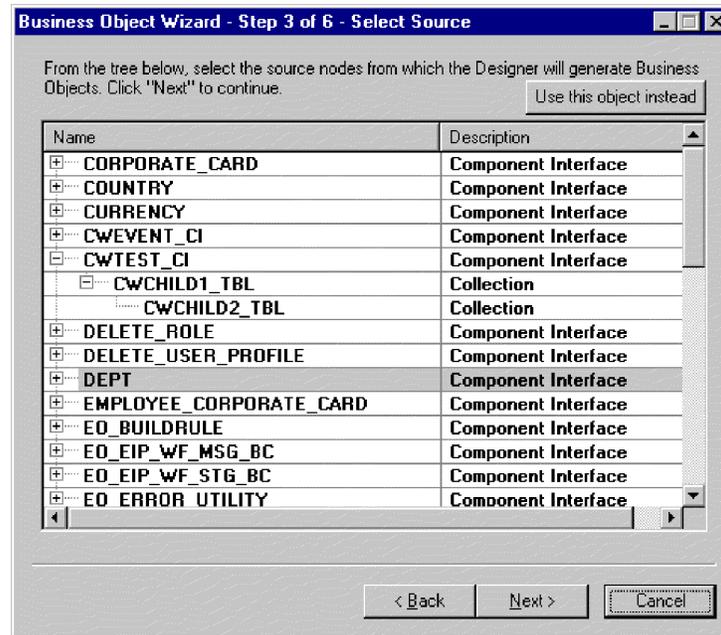


図 12. ノードが展開されたコンポーネント・インターフェースのツリー

必要なすべてのコレクションとともに、必要なすべてのコンポーネント・インターフェースを選択して、「次へ」をクリックします。

オブジェクトの選択の確認

生成されたビジネス・オブジェクト定義に関連するコンポーネント・インターフェースとコレクションをすべて特定すると、Business Object Designer により、選択したオブジェクトのみを含むダイアログ・ボックスが表示されます。図 13 にこのダイアログ・ボックスを示します。

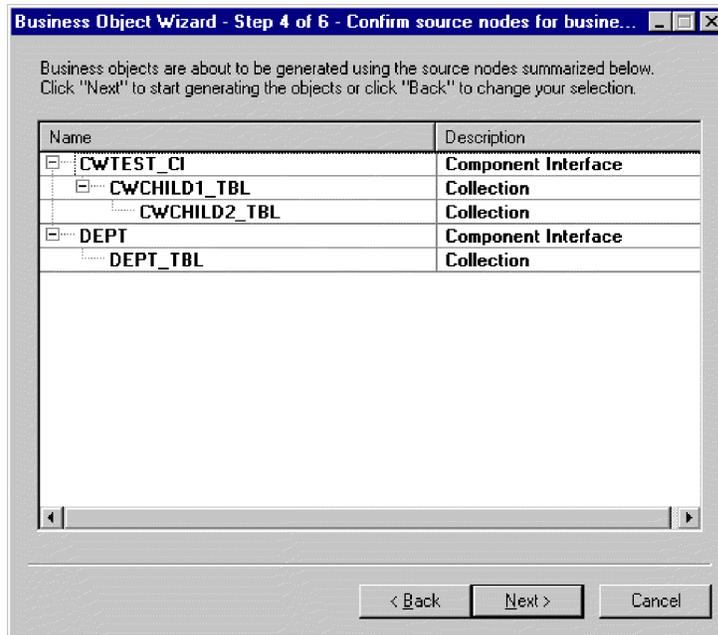


図 13. オブジェクトの選択の確認

このウィンドウには、以下のオプションが表示されます。

- 選択を確認するには、「次へ」をクリックします。
- 選択に誤りがあった場合には、「戻る」をクリックして、直前のウィンドウに戻り、必要な変更を加えます。選択が正しい場合には、「次へ」をクリックします。

定義の生成

コンポーネント・インターフェースとコレクションを確認すると、その次のダイアログ・ボックスで Business Object Designer が定義を生成することが通知されます。多くの数のコンポーネント・インターフェースが選択されると、この生成ステップは時間がかかることがあります。

図 14 にこのダイアログ・ボックスを示します。

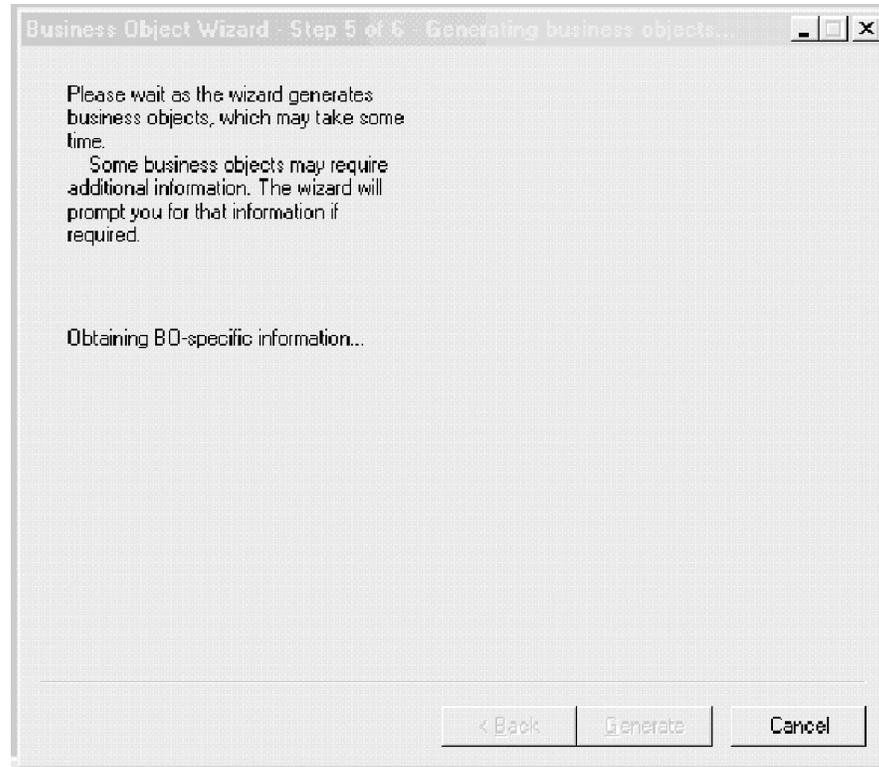


図 14. 定義の生成

追加情報の入力

PeopleSoftODA に追加情報が必要な場合、Business Object Designer により、情報の入力を要求する「BO プロパティ」ウィンドウが表示されます。図 15 にこのダイアログ・ボックスを示します。

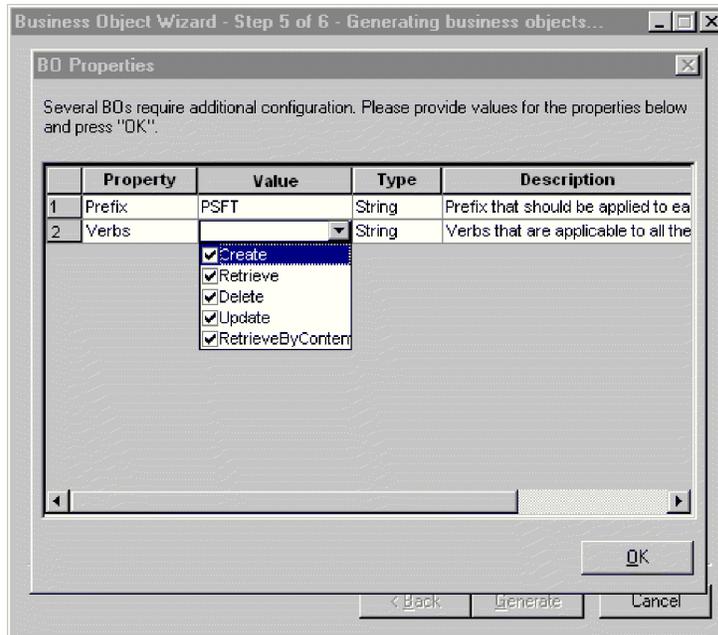


図 15. 追加情報の入力

「BO プロパティ」ウィンドウで、次の情報を入力または変更します。

- **プレフィックス:** ビジネス・オブジェクト名を固有にするために、その名前の前に付加するテキスト。「エージェントの構成」ウィンドウ (79 ページの図 11) で *DefaultBOPrefix* プロパティに対して入力した値が適切であれば、ここでこの値を変更する必要はありません。
- **動詞:** 「値」フィールドをクリックし、ポップアップ・メニューから 1 つ以上の動詞を選択します。これらはビジネス・オブジェクトでサポートされる動詞です。

注: 「BO プロパティ」ダイアログ・ボックスのフィールドに複数の値が存在する場合、ダイアログ・ボックスが最初に表示されるときにはフィールドが空に見えます。フィールド内をクリックして、その値のドロップダウン・リストを表示します。

定義の保管

「BO プロパティ」ダイアログ・ボックスで必要なすべての情報を指定し、「OK」をクリックすると、Business Object Designer にウィザードの最終ダイアログ・ボックスが表示されます。このダイアログ・ボックスで、定義をサーバー (ICS が統合ブローカーの場合) かまたはファイル (すべての統合ブローカー) に保管するか、あるいは Business Object Designer で定義を開いて編集できます。この詳細と、さらに変更する方法については、「ビジネス・オブジェクト開発ガイド」を参照してください。

図 16 にこのダイアログ・ボックスを示します。

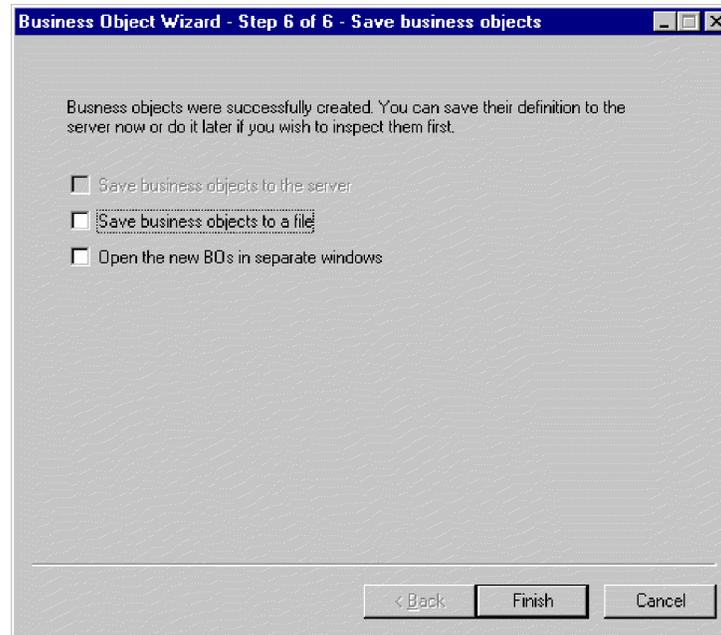


図 16. ビジネス・オブジェクト定義の保管

生成された定義の内容

PeopleSoftODA が生成するビジネス・オブジェクト定義には、次のものがあります。

- 指定されたコンポーネント・インターフェース内の各プロパティの単純属性。
- コンポーネント・インターフェースの階層表現で選択された各コレクションの子ビジネス・オブジェクトの配列を表す属性。
- 「BO プロパティ」ウィンドウ (84 ページの図 15 を参照) で指定された動詞。
- 以下のアプリケーション固有情報。
 - ビジネス・オブジェクト・レベルでの情報
 - 各属性に関する情報

このセクションでは、以下について説明します。

- 『ビジネス・オブジェクト・レベルのプロパティ』
- 86 ページの『属性プロパティ』

ビジネス・オブジェクト・レベルのプロパティ

PeopleSoftODA は、ビジネス・オブジェクト・レベルで次の情報を生成します。

- ビジネス・オブジェクトの名前。
- バージョン (デフォルトは 1.0.0)。
- アプリケーション固有情報。

ビジネス・オブジェクト・レベルでは、アプリケーション固有情報のフォーマットはコロン (:) 区切り文字で区切られた 4 つのパラメーターで構成されます。フォーマットは、次のとおりです。

```
cIName=ComponentInterface:EFFDT=[true|false]:setInteractiveMode=[true|false]:
GetHistoryItems=[true|false]:SetEditHistoryItems=[true|false]
```

表 10 で、これらのパラメーターについて説明します。

表 10. ビジネス・オブジェクト・レベルの *AppSpecificInfo*

AppSpecificInfo パラメーター	説明
ComponentInterface	PeopleSoft アプリケーションに定義されたコンポーネント・インターフェースの名前を指定します。
setInteractiveMode	コネクターが変更をアプリケーション・サーバーに送信する時期を判別します。
GetHistoryItems	コネクターが検索するデータ量を決定します。
EFFDT	ビジネス・オブジェクト (または子ビジネス・オブジェクト) が有効期限を使用するかどうかを指定します。
SetEditHistoryItems	使用可能な編集およびデータ履歴の保管。有効期限フィールドでのみ使用されます。

重要: PeopleSoftODA は、ComponentInterface プロパティの値を指定するための定義を生成したコンポーネント・インターフェースの名前を使用します。他のプロパティの値は指定しません。ビジネス・オブジェクト定義を変更して、残りのプロパティの値を指定する必要があります。これらのプロパティの詳細については、67 ページの『ビジネス・オブジェクト・レベルのアプリケーション固有情報』を参照してください。ビジネス・オブジェクト定義の変更方法については、「ビジネス・オブジェクト開発ガイド」を参照してください。

属性プロパティ

このセクションでは、各属性について PeopleSoftODA が生成するプロパティについて説明します。属性の詳細については、64 ページの『ビジネス・オブジェクトの属性プロパティ』を参照してください。

Name プロパティ

PeopleSoftODA は、対応するコンポーネント・インターフェース内のプロパティから属性の名前を派生させます。

Data Type プロパティ

属性のタイプを設定する場合、PeopleSoftODA は、表 11 に示すように、プロパティのデータ型を対応するビジネス・オブジェクトのデータ型に変換します。

表 11. データ型の対応

PeopleSoft	ビジネス・オブジェクト	長さ
String	String	データ型で指定される長さ
Boolean	Boolean	
Collection	Object	

表 11. データ型の対応 (続き)

PeopleSoft	ビジネス・オブジェクト	長さ
Float	Float	
Number	Integer	

注: プロパティのデータ型が表 11 に示す中のいずれでもない場合、PeopleSoftODA はプロパティをスキップして、プロパティを処理できないことを示すメッセージを表示します。

Cardinality プロパティ

PeopleSoftODA はすべての単純属性のカーディナリティーを 1 に設定します。また、子ビジネス・オブジェクトの配列を表すすべての属性のカーディナリティーを n に設定します。

MaxLength プロパティ

PeopleSoftODA は、ストリングのデフォルトの長さを 255 文字に設定します。その他のデータ型の場合はすべて、対応するビジネス・オブジェクトのデータ型の標準最大長を使用します。

IsKey プロパティ

プロパティがコンポーネント・インターフェース内の CreateKey の場合、PeopleSoftODA はこのプロパティを true に設定します。プロパティがコンポーネント・インターフェース内の GetKey の場合、PeopleSoftODA はこのプロパティに false を設定し、属性の AppSpecificInfo パラメーターに GetKey=true を設定します。

IsForeignKey プロパティ

PeopleSoftODA はこのプロパティを false に設定します。Business Object Designer で設定を変更することができます。

IsRequired プロパティ

PeopleSoftODA は一部のキーを内部的に生成するので、このプロパティを常に false に設定します。Business Object Designer で設定を変更することができます。

AppSpecificInfo プロパティ

属性アプリケーション固有情報は、コロン (:) 区切り文字でお互いに区切られる 5 セットの名前値パラメーターです。垂直バー (|) は、オプション・セットのメンバーを区切ります。フォーマットは、次のとおりです。

```
get=getFieldName:set=setFieldName:UID=[true|false]:GetKey=[true|false]:KeepRelationship=[true|false]
```

PeopleSoftODA は、表 12 で説明するように、属性に関連するプロパティのみを生成します。複数のパラメーターを生成する場合は、コロンでパラメーターを区切ります。

表 12. PeopleSoftODA で生成される属性 AppSpecificInfo

AppSpecificInfo パラメーター	説明
GetKey=true	PeopleSoftODA は、GetKey として定義されるコンポーネント・インターフェースのプロパティに対応する属性に関してのみ、このパラメーターを生成します。コネクタはこのような属性の値を使用して、コンポーネント・インターフェースのインスタンスを検索します。
get=getPropertyName	PropertyName の場合、PeopleSoftODA は属性に関連するコンポーネント・インターフェースのプロパティの名前を置換します。コンポーネント・インターフェースのプロパティに対応するすべての単純属性に関して、このパラメーターを生成します。コネクタはこの方法で、更新された属性の値を検索します。
get=getCollectionName	CollectionName の場合、PeopleSoftODA は属性に関連するコンポーネント・インターフェースのコレクションの名前を置換します。コンポーネント・インターフェースのコレクションに対応する子ビジネス・オブジェクトの配列を表す属性のすべてに関して、このパラメーターを生成します。コネクタはこの方法で、コレクションの値を検索します。
set=setPropertyName	PropertyName の場合、PeopleSoftODA は属性に関連するコンポーネント・インターフェースのプロパティの名前を置換します。コンポーネント・インターフェースのプロパティに対応するすべての単純属性に関して、このパラメーターを生成します。コネクタはこの方法で、属性の値を更新します。

注: Business Object Designer で、追加の AppSpecificInfo パラメーターを設定できます。これらのパラメーターの詳細については、69 ページの『属性レベルのアプリケーション固有情報』を参照してください。定義の変更方法については、「ビジネス・オブジェクト開発ガイド」を参照してください。表 12 で説明されているパラメーターの使用例については、『ビジネス・オブジェクト定義ファイルのサンプル』を参照してください。

動詞

注: PeopleSoftODA は、「BO プロパティ」ウィンドウ (84 ページの図 15 を参照) で指定される動詞を生成します。

ビジネス・オブジェクト定義ファイルのサンプル

この製品には、3 つのサンプル・ビジネス・オブジェクト定義ファイルが付属しています。

- BO_Psft_DEPT
- BO_PsftEmployee
- SavePostChange

BO_Psft_DEPT ビジネス・オブジェクト

次に示す例は BO_Psft DEPT ビジネス・オブジェクトです。

```
[BusinessObjectDefinition] Name = DeptTbl Version = 1.0.0 AppSpecificInfo =
CiName=DEPT [Attribute] Name = Company Type = String
MaxLength = 255 IsKey = true IsForeignKey = false IsRequired =
false AppSpecificInfo = get=getCompany:set=setCompany
IsRequiredServerBound = false [End] [Attribute] Name =
BudgetLvl Type = String MaxLength = 1 IsKey = false
IsForeignKey = false IsRequired = false AppSpecificInfo =
get=getBudgetLvl:set=setBudgetLvl IsRequiredServerBound = false
[End] [Attribute] Name = Descr Type = String MaxLength = 1
IsKey = false IsForeignKey = false IsRequired = false
AppSpecificInfo = get=getDescr:set=setDescr IsRequiredServerBound =
false [End] [Attribute] Name = DescrShort Type = String
MaxLength = 1 IsKey = false IsForeignKey = false IsRequired =
false AppSpecificInfo = get=getDescrshort:set=setDescrshort
IsRequiredServerBound = false [End] [Attribute] Name =
ObjectEventId Type = String MaxLength = 255 IsKey = false
IsForeignKey = false IsRequired = false IsRequiredServerBound =
false [End] [Verb] Name = Create [End] [Verb] Name
= Delete [End] [Verb] Name = Retrieve [End] [Verb]
Name = Update [End] [End] [BusinessObjectDefinition] Name = Psft_dept
Version = 1.0.0 AppSpecificInfo = CiName=DEPT [Attribute] Name =
Deptid Type = String MaxLength = 255 IsKey = true
IsForeignKey = false IsRequired = true AppSpecificInfo =
get=getDeptid:set=setDeptid:GetKey=true IsRequiredServerBound = false
[End] [Attribute] Name = Setid Type = String MaxLength
= 1 IsKey = true IsForeignKey = false IsRequired = true
AppSpecificInfo = get=getSetid:set=setSetid:GetKey=true
IsRequiredServerBound = false [End] [Attribute] Name =
StartDate Type = Date MaxLength = 1 IsKey = false
IsForeignKey = false IsRequired = true AppSpecificInfo =
get=getStartDate:set=setStartDate IsRequiredServerBound = false
[End] [Attribute] Name = DptTbl Type = DeptTbl
ContainedObjectVersion = 1.0.0 Relationship = Containment
Cardinality = n MaxLength = 1 IsKey = false IsForeignKey =
false IsRequired = true AppSpecificInfo =
get=getDeptTbl:KEEPRELATIONSHIP=true IsRequiredServerBound = false
[End] [Attribute] Name = ObjectEventId Type = String
MaxLength = 255 IsKey = false IsForeignKey = false IsRequired =
false IsRequiredServerBound = false [End] [Verb] Name =
Create [End] [Verb] Name = Delete [End] [Verb] Name
= Retrieve [End] [Verb] Name = Update [End] [End]
```

BO_PsftEmployee ビジネス・オブジェクト

次に示す例は BO_PsftEmployee ビジネス・オブジェクトです。

```

[BusinessObjectDefinition] Name = PSFTEmployee Version = 1.0.0
AppSpecificInfo = cIName=Emp      [Attribute]      Name = EMPID      Type =
String      Cardinality = 1      MaxLength = 255      IsKey = true
IsForeignKey = false      IsRequired = true      AppSpecificInfo =
get=getEmpId:set=setEmpId:keepRelationship=false:uid=true:
findKey=true:getKey=true      IsRequiredServerBound = false      [End]
[Attribute]      Name = EMPL_RCD      Type = String      Cardinality = 1
MaxLength = 1      IsKey = true      IsForeignKey = false      IsRequired =
true      AppSpecificInfo =
get=getEmpRcd:set=setEmpRcd:keepRelationship=false:uid=true:
findKey=true:getKey=true      IsRequiredServerBound = false      [End]
[Attribute]      Name = NAME      Type = String      Cardinality = 1
MaxLength = 1      IsKey = true      IsForeignKey = false      IsRequired =
true      AppSpecificInfo =
get=getName:set=setName:keepRelationship=false:uid=false:
findKey=true:getKey=false:createKey=false      IsRequiredServerBound = false
[End]      [Attribute]      Name = LAST_NAME_SRCH      Type = String
Cardinality = 1 MaxLength = 1      IsKey = true      IsForeignKey = false
IsRequired = true      AppSpecificInfo =
get=getLastNameSrch:set=setLastNameSrch:keepRelationship=false:uid=false:
findKey=true:getKey=false:createKey=false      IsRequiredServerBound =
false      [End]      [Attribute]      Name = NAME_AC      Type = String
Cardinality = 1      MaxLength = 1      IsKey = true      IsForeignKey = false
IsRequired = true      AppSpecificInfo =
get=getNameAc:set=setNameAc:keepRelationship=false:uid=false:
findKey=true:getKey=false:createKey=false      IsRequiredServerBound = false
[End]      [Attribute]      Name = PER_STATUS      Type = String
Cardinality = 1      MaxLength = 1      IsKey = true      IsForeignKey = false
IsRequired = true      AppSpecificInfo =
get=getPerStatus:set=setPerStatus:keepRelationship=false:uid=false:
findKey=true:getKey=false:createKey=false      IsRequiredServerBound = false
[End]      [Attribute]      Name = EMPLID_0      Type = String
Cardinality = 1      MaxLength = 1      IsKey = false      IsForeignKey =
false      IsRequired = false      AppSpecificInfo =
get=getEmpId0:set=setEmpId0:keepRelationship=false:uid=false:
findKey=false:getKey=false:createKey=false      IsRequiredServerBound =
false      [End]      [Attribute]      Name = ORIG_HIRE_DT      Type = String
Cardinality = 1      MaxLength = 1      IsKey = false      IsForeignKey =
false      IsRequired = false      AppSpecificInfo =
get=getOrigHireDt:set=setOrigHireDt:keepRelationship=false:uid=false:
findKey=false:getKey=false:createKey=false      IsRequiredServerBound =
false      [End]      [Attribute]      Name = SEX      Type = String
Cardinality = 1      MaxLength = 1      IsKey = false      IsForeignKey =
false      IsRequired = true      AppSpecificInfo =
get=getSex:set=setSex:keepRelationship=false:uid=false:findKey=false:
getKey=false:createKey=false      IsRequiredServerBound = false      [End]
[Attribute]      Name = BIRTHDATE      Type = String      Cardinality = 1
MaxLength = 1      IsKey = false      IsForeignKey = false      IsRequired

```

```

= false      AppSpecificInfo =
get=getBirthdate:set=setBirthdate:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name =
FT_STUDENT      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = true
AppSpecificInfo = get=getFtStudent:set=setFtStudent:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name =
BENEFIT_RCD_NBR      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getBenefitRcdNbr:set=setBenefitRcdNbr:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name =
HOME_HOST_CLASS      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = true
AppSpecificInfo = get=getHomeHostClass:set=setHomeHostClass:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name =
HIRE_DT      Type = String      Cardinality = 1      MaxLength = 1      IsKey =
false      IsForeignKey = false      IsRequired = false      AppSpecificInfo =
get=getHireDt:set=setHireDt:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name =
CMPNY_SENIORITY_DT      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getCmpnySeniorityDt:
set=setCmpnySeniorityDt:keepRelationship=false:uid=false:
findKey=false:getKey=false:createKey=false      IsRequiredServerBound =
false      [End]      [Attribute]      Name = SERVICE_DT      Type = String
Cardinality = 1      MaxLength = 1      IsKey = false      IsForeignKey =
false      IsRequired = false      AppSpecificInfo =
get=getServiceDt:set=setServiceDt:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name =
PROF_EXPERIENCE_DT      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getProfExperienceDt:
set=setProfExperienceDt:keepRelationship=false:uid=false:
findKey=false:getKey=false:createKey=false      IsRequiredServerBound =
false      [End]      [Attribute]      Name = LAST_VERIFICATN_DT      Type =
String      Cardinality = 1      MaxLength = 1      IsKey = false
IsForeignKey = false      IsRequired = false      AppSpecificInfo =
get=getLastVerificatnDt:set=setLastVerificatnDt:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name =
EXPECTED_RETURN_DT      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = false

```

```

AppSpecificInfo = get=getExpectedReturnDt:set=
setExpectedReturnDt:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name =
LAST_DATE_WORKED      Type = String      Cardinality = 1      MaxLength = 1
      IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getLastDateWorked:
set=setLastDateWorked:keepRelationship=false:uid=false:
findKey=false:getKey=false:createKey=false      IsRequiredServerBound =
false      [End]      [Attribute]      Name = LAST_INCREASE_DT      Type =
String      Cardinality = 1      MaxLength = 1      IsKey = false
IsForeignKey = false      IsRequired = false      AppSpecificInfo =
get=getLastIncreaseDt:      set=setLastIncreaseDt:keepRelationship=false:
      uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name =
OWN_5PERCENT_CO      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = true
AppSpecificInfo = get=getOwn5percentCo:
set=setOwn5percentCo:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name =
BUSINESS_TITLE      Type = String      Cardinality = 1      MaxLength = 1
IsKey = true      IsForeignKey = false      IsRequired = true
AppSpecificInfo = get=getBusinessTitle:
set=setBusinessTitle:keepRelationship=false:
uid=false:findKey=false:getKey=true:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name =
REPORTS_TO      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getReportsTo:set=setReportsTo:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
      IsRequiredServerBound = false      [End]      [Attribute]      Name =
SUPERVISOR_ID      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getSupervisorId:set=setSupervisorId:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
      IsRequiredServerBound = false      [End]      [Attribute]      Name =
PROBATION_DT      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getProbationDt:set=setProbationDt:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
      IsRequiredServerBound = false      [End]      [Attribute]      Name =
SECURITY_CLEARANCE      Type = String      Cardinality = 1      MaxLength = 1
      IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getSecurityClearance:
set=setSecurityClearance:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name = PHONE

```

```

    Type = String      Cardinality = 1      MaxLength = 1      IsKey = false
    IsForeignKey = false      IsRequired = false      AppSpecificInfo =
get=getPhone:set=setPhone:      keepRelationship=false:uid=false:
findKey=false:getKey=false:createKey=false      IsRequiredServerBound =
false      [End]      [Attribute]      Name = TIME_RPT_LOCK      Type = String
    Cardinality = 1      MaxLength = 1      IsKey = false      IsForeignKey =
false      IsRequired = false      AppSpecificInfo =
get=getTimeRptLock:set=setTimeRptLock:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false      [End]      [Attribute]      Name =
JOB_REPORTING      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = true
AppSpecificInfo = get=getJobReporting:set=setJobReporting:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false      [End]      [Attribute]      Name =
DED_TAKEN      Type = String      Cardinality = 1      MaxLength = 1      IsKey
= false      IsForeignKey = false      IsRequired = true      AppSpecificInfo
= get=getDedTaken:set=setDedTaken:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false      [End]      [Attribute]      Name =
DED_SUBSET_ID      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getDedSubsetId:set=setDedSubsetId:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false      [End]      [Attribute]      Name =
CAN_ABORIGINAL      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getCanAboriginal:set=setCanAboriginal:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false      [End]      [Attribute]      Name =
CAN_VISBL_MINORITY      Type = String      Cardinality = 1      MaxLength = 1
    IsKey = false      IsForeignKey = false      IsRequired = true
AppSpecificInfo = get=getCanVisblMinority:set=setCanVisblMinority:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false      [End]      [Attribute]      Name =
CURRENT_SEQ      Type = String      Cardinality = 1      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getCurrentSeq:set=setCurrentSeq:
keepRelationship=false:uid=false:findKey=false:getKey=false:createKey=false
    IsRequiredServerBound = false      [End]      [Attribute]      Name =
PERS_DATA_EFFDT      Type = PERS_DATA_EFFDT      ContainedObjectVersion =
1.0.0      Relationship = Containment      Cardinality = n      MaxLength = 1
    IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getPersDataEffdt:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name =
EMAIL_ADDRESSES      Type = EMAIL_ADDRESSES      ContainedObjectVersion =
1.0.0      Relationship = Containment      Cardinality = n      MaxLength = 1

```

```

        IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getEmailAddresses:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name =
PERSONAL_PHONE      Type = PERSONAL_PHONE      ContainedObjectVersion = 1.0.0
        Relationship = Containment      Cardinality = n      MaxLength = 1
IsKey = false      IsForeignKey = false      IsRequired = false
AppSpecificInfo = get=getPersonalPhone:keepRelationship=false:
uid=false:findKey=false:getKey=false:createKey=false
IsRequiredServerBound = false      [End]      [Attribute]      Name = PERS_NID
        Type = PERS_NID      ContainedObjectVersion = 1.0.0      Relationship =
Containment      Cardinality = n      MaxLength = 1      IsKey = false
IsForeignKey = false      IsRequired = false      IsRequiredServerBound =
false      [End]      [Attribute]      Name = JOB      Type = JOB
ContainedObjectVersion = 1.0.0      Relationship = Containment
Cardinality = n      MaxLength = 1      IsKey = false      IsForeignKey =
false      IsRequired = false      IsRequiredServerBound = false      [End]
        [Attribute]      Name = ObjectEventId      Type = String      MaxLength =
255      IsKey = false      IsForeignKey = false      IsRequired = false
IsRequiredServerBound = false      [End]      [Verb]      Name = Create
[End]      [Verb]      Name = Delete      [End]      [Verb]      Name = Retrieve
[End]      [Verb]      Name = Update      [End] [End]

```

SavePostChange ビジネス・オブジェクトの例

次の例は、関数呼び出し `cw_publish_event` を示す **SavePostChange** ビジネス・オブジェクトです。

```

/* Place this code in Component's SavePostChg() and define the four
parameters used in the function call */ Declare Function cw_publish_event
PeopleCode FUNCLIB_CW.CW_EVENT_NOT FieldFormula; Component string &BONAME1;
Component string &KEYLIST1; Component number &CWPRORITY1; Component string
&CONNID; &BONAME1 = "Psft_Dept"; &KEYLIST1 =
"DEPT_TBL.SETID:DEPT_TBL.DEPTID"; &CWPRORITY1 = 2; &CONNID = "PeopleSoft
Connector"; /* Check if Component Changed before calling function */ If
ComponentChanged() And %UserId <> "CW" Then /* Publish this event to
the CrossWorlds CW_EVENT_TBL for polling */ cw_publish_event(&BONAME1,
&KEYLIST1, &CWPRORITY1, &CONNID); End-If;

```

次の例は、関数呼び出し `cw_publish_future_dated_events` を示す **SavePostChange** ビジネス・オブジェクトです。

```

/* Place this code in Component's SavePostChg() and define the four
parameters used in the function call */ Declare Function cw_publish_event
PeopleCode FUNCLIB_CW.CW_EVENT_NOT FieldFormula; Component string &BONAME1;
Component string &KEYLIST1; &EFFDATE="DEPT_TBL_.STARTDATE"; Component
number &CWPRORITY1; Component string &CONNID; &BONAME1 = "Psft_Dept";
&KEYLIST1 = "DEPT_TBL.SETID:DEPT_TBL.DEPTID"; &CWPRORITY1 = 2; &CONNID =
"PeopleSoft Connector"; /* Check if Component Changed before calling

```

```
function */ If ComponentChanged() And %UserId <> "CW" Then /* Publish
this event to the CrossWorlds CW_EVENT_TBL for polling */
cw_publish_future_dated_events(&BONAME1, &KEYLIST1, &CWPRORITY1, &CONNID,
&EFFDATEW); End-If;
```

ビジネス・オブジェクト定義への情報の追加

PeopleSoftODA が作成するビジネス・オブジェクト定義内の情報を変更する必要がある場合があります。例えば、不要な属性を手動で除去したり、ビジネス・オブジェクト・レベルでのアプリケーション固有情報の `getHistoryItems` や `setInteractiveMode` パラメーターのデフォルト値を変更したり、属性アプリケーション固有情報の必要パラメーターを追加したりする必要があります。詳細については、45 ページの『第 4 章 コネクター用のビジネス・オブジェクトについて』を参照してください。

ビジネス・オブジェクト定義を検査または変更するには、`Business Object Designer` または `テキスト・エディター` を使用します。改訂された定義をリポジトリに再ロードするには、`Business Object Designer` を使用します。あるいは、`WebSphere InterChange Server (ICS)` が統合ブローカーの場合、`repos_copy` コマンドを使用して定義をリポジトリにロードすることができます。`WebSphere MQ Integrator` のブローカーが統合ブローカーの場合は、システム・コマンドを使用してファイルをリポジトリ・ディレクトリにコピーできます。

第 6 章 コネクタの実行

この章では、コネクタの始動、停止、およびアップグレード方法について説明します。

- 『コネクタの始動』
- 98 ページの『コネクタの停止』
- 99 ページの『コネクタのアップグレード』

コネクタの始動

コネクタは、**コネクタ始動スクリプト**を使用して明示的に始動する必要があります。始動スクリプトは、次に示すようなコネクタのランタイム・ディレクトリに存在していなければなりません。

```
ProductDir¥connectors¥connName
```

ここで、*connName* はコネクタを示します。始動スクリプトの名前は、表 13 に示すように、オペレーティング・システム・プラットフォームによって異なります。

表 13. コネクタの始動スクリプト

オペレーティング・システム	始動スクリプト
UNIX ベースのシステム	connector_manager_connName
Windows	start_connName.bat

コネクタ始動スクリプトは、以下に示すいずれかの方法で起動することができます。

- Windows システムで「スタート」メニューから。

「プログラム」>「IBM WebSphere Business Integration Adapters」>「アダプター」>「コネクタ」を選択します。デフォルトでは、プログラム名は「IBM WebSphere Business Integration Adapters」となっています。ただし、これはカスタマイズすることができます。あるいは、ご使用のコネクタへのデスクトップ・ショートカットを作成することもできます。

- コマンド行から。

– Windows システム:

```
start_connName connName brokerName [-cconfigFile ]
```

– UNIX ベースのシステム:

```
connector_manager_connName -start
```

ここで、*connName* はコネクタの名前であり、*brokerName* は以下のようにご使用の統合ブローカーを表します。

- WebSphere InterChange Server の場合は、*brokerName* に ICS インスタンスの名前を指定します。

- WebSphere Message Brokers (WebSphere MQ Integrator, WebSphere MQ Integrator Broker, または WebSphere Business Integration Message Broker) または WebSphere Application Server の場合は、*brokerName* にブローカーを示すストリングを指定します。

注: Windows システム上の WebSphere Message Broker または WebSphere Application Server の場合は、*-c* オプションに続いてコネクタ構成ファイルの名前を指定しなければなりません。ICS の場合は、*-c* はオプションです。

- Adapter Monitor から (WebSphere Business Integration Adapters 製品のみ)。Adapter Monitor は System Manager 始動時に起動されます。

このツールを使用して、コネクタのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。

- System Monitor から (WebSphere InterChange Server 製品のみ)。

このツールを使用して、コネクタのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。

- Windows システムでは、Windows サービスとして始動するようにコネクタを構成することができます。この場合、Windows システムがブートしたとき (自動サービスの場合)、または Windows サービス・ウィンドウを通じてサービスを始動したとき (手動サービスの場合) に、コネクタが始動します。

コマンド行の始動オプションなどのコネクタの始動方法の詳細については、以下の資料のいずれかを参照してください。

- WebSphere InterChange Server については、「システム管理ガイド」を参照してください。
- WebSphere Message Brokers については、「*WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド*」を参照してください。
- WebSphere Application Server については、「*アダプター実装ガイド (WebSphere Application Server)*」を参照してください。

コネクタの停止

コネクタを停止する方法は、以下に示すように、コネクタが始動された方法によって異なります。

- コマンド行からコネクタを始動した場合は、コネクタ始動スクリプトを用いて、以下の操作を実行します。
 - Windows システムでは、始動スクリプトを起動すると、そのコネクタ用の別個の「コンソール」ウィンドウが作成されます。このウィンドウで、「Q」と入力して Enter キーを押すと、コネクタが停止します。
 - UNIX ベースのシステムでは、コネクタはバックグラウンドで実行されるため、別ウィンドウはありません。代わりに、次のコマンドを実行してコネクタを停止します。

```
connector_manager_connName -stop
```

ここで、*connName* はコネクタの名前です。

- Adapter Monitor から (WebSphere Business Integration Adapters 製品のみ)。
Adapter Monitor は System Manager 始動時に起動されます。

このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。

- System Monitor から (WebSphere InterChange Server 製品のみ)

このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。

- Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムのシャットダウン時に、コネクターは停止します。

コネクターのアップグレード

WebSphere Business Integration Adapter for PeopleSoft の以前のバージョンをインストールしている場合、それを削除してから、このコネクターをインストールする必要があります。

また、PeopleSoft アップグレードおよびフィックスのプロジェクト標準に従って、コネクターに関して PeopleSoft で行った以前のカスタマイズをすべてアップグレードまたは削除します。コネクターの各バージョンは、現行のすべての PeopleSoft オブジェクトやイベント通知に必要なコードなどの Application Designer プロジェクト (メンテナンス・プロジェクト) に組み込まれています。プロジェクトはまた、プロジェクト・プロパティの詳細部分のオブジェクトに対して行われたアダプター固有のフィックスや機能強化もリストします。このプロジェクトをデータベースにインポートまたはコピーするには、一般的なプロジェクト標準に従います。メニュー、ページ、レコード、またはフィールドが不要になった場合、削除が必要になることがあります。

新規のコネクターをインストールするには、13 ページの『第 2 章 コネクターのインストール』の説明に従ってください。

第 7 章 トラブルシューティングとエラー処理

この章では、IBM WebSphere Business Integration Adapter for PeopleSoft を始動または実行する際に発生する可能性がある問題について説明します。

この章は、以下のセクションから構成されています。

- 『始動時の問題』
- 『始動時の問題 (WebSphere InterChange Server ブローカーのみ)』
- 102 ページの『始動時の問題 (WebSphere MQ Integrator Broker のみ)』
- 102 ページの『処理時の問題』
- 103 ページの『マッピング (ICS 統合ブローカーのみ)』
- 103 ページの『エラー処理とロギング』
- 104 ページの『アプリケーションへの接続の切断』

始動時の問題

コネクターを始動する際に問題が発生した場合は、次のことを行ってください。

- psadmin ユーティリティを使用して、PeopleSoft アプリケーション・サーバーが稼働していることを確認します。
- Application Designer の Test Component Interface ユーティリティを使用して、コンポーネント・インターフェースがオンラインで動作していることを確認します。
- PeopleSoft の API を再ビルドおよび再コンパイルします。詳細については、52 ページの『API の生成』を参照してください。
- PeopleSoft セキュリティーが正しく設定され、コンポーネント・インターフェースがアクセス権リストにリストされ、またコネクターのユーザー・アカウントにタイムアウトが設定されていることを確認します。詳細については、16 ページの『ユーザー・アカウントのタイムアウトの構成』を参照してください。
- コネクターのユーザー・アカウントに行レベルのセキュリティが設定されていることを確認します。
- PeopleSoft 構成マネージャーの「トレース」タブの関連するオプションを選択して、API をトレースします。

始動時の問題 (WebSphere InterChange Server ブローカーのみ)

コネクターで WebSphere InterChange MQ Integrator Broker を統合ブローカーとして使用すると、コネクターの始動時に問題が発生します。

- ICS が稼働していることを確認します。

UNIX の場合、次のコマンドを入力します。

```
ICS_manager -stat
```

Windows の場合、`start_connector` コマンドの後の 2 番目のパラメーターに、InterChange Server の名前を指定します。例えば、次のようになります。

```
start_PeopleSoftConnectorName InterChangeServerName
```

ここで、`InterChangeServerName` は ICS インスタンスの名前です。

- 101 ページの『始動時の問題』の指示に従います。

始動時の問題 (WebSphere MQ Integrator Broker のみ)

コネクタで MQ Integrator Broker を統合ブローカーとして使用すると、コネクタ一の始動時に問題が発生します。

- WebSphere MQ Integrator Broker が稼働していることを確認します。

UNIX の場合、次のコマンドを入力します。

```
ICS_manager -stat
```

Windows の場合、`start_connector` コマンドに、コネクタとブローカーの名前、およびコネクタの構成ファイルのパスとファイル名を指定する必要があります。例えば、次のようになります。

```
start_PeopleSoftConnectorName BrokerName-cConfigFileName
```

ここで、`BrokerName` は WebSphere MQ Integrator Broker の名前、`ConfigFileName` にコネクタの構成ファイルのパスとファイル名を指定します。

- 101 ページの『始動時の問題』の指示に従います。

処理時の問題

コネクタの処理中にエラーが発生した場合は、簡単な Java プログラムを作成し、IDE を使用して以下のテストを行います。

1. アプリケーションへの接続。
2. `Get()`、`Create()`、および `Find()` メソッド。
3. 対応するコンポーネント・インターフェースの各レベルからのプロパティ値の検索。

このテストで、PeopleSoft API とコネクタの間で発生する問題を切り分けることができます。つまり、テスト・プログラムで問題が発生しなければ、コネクタのコードに問題がある可能性があります。

PeopleSoft で、テスト用 Java プログラムのテンプレートを生成します。テンプレートを生成するには、次の手順で行います。

1. 問題のコンポーネント・インターフェースを開く。
2. ウィンドウ内の任意の場所を右クリックして、ポップアップ・メニューから「Java テンプレートの生成 (Generate Java Template)」を選択する。

マッピング (ICS 統合ブローカーのみ)

ビジネス・オブジェクトがマップされていないかまたはマッピングが呼び出されていない場合は、適切なディレクトリーにマップがインストールされていることを確認します。

エラー処理とロギング

コネクターは、ビジネス・オブジェクトと動詞の現在の処理がエラーとなる条件を検出すると、常にエラー・メッセージをログに記録します。このようなエラーが発生すると、コネクターは失敗したビジネス・オブジェクトを受け取ったままにテキスト表現で出力します。構成に応じて、コネクターのログ・ファイルか、または標準出力ストリームにテキストを書き込みます。エラーの発生元を判別するための補助資料としてこのテキストを使用できます。

エラー・タイプ

表 14 で、コネクターが各トレース・レベルで出力するトレース・メッセージのタイプを説明します。これらは、Java コネクター実行ラッパーや IBM MQSeries メッセージ・インターフェースなどの、WebSphere Business Integration システムのアーキテクチャーで出力されるトレース・メッセージに追加されるメッセージです。

表 14. PeopleSoft トレース・メッセージ

トレース・レベル	トレース・メッセージ
レベル 0	コネクターのバージョンを識別するメッセージ。このレベルでは、他のトレースは実行されません。これはデフォルト値です。
レベル 1	<ul style="list-style-type: none">• 状況メッセージ。• 処理される各ビジネス・オブジェクトの識別 (キー) 情報を指定するメッセージ。• メッセージは、pollForEvents メソッドが実行されるたびにデリバリーされます。
レベル 2	<ul style="list-style-type: none">• コネクターがビジネス・オブジェクトを処理時に検出または検索する配列や子ビジネス・オブジェクトなどの情報が格納されるビジネス・オブジェクト・ハンドラー・メッセージ。• ビジネス・オブジェクトが gotApp1Event() かまたは consumeSync() から統合ブローカーにポストされる際に、常にログに記録されるメッセージ。• ビジネス・オブジェクトを要求どおりに受け取ったことを示すメッセージ。
レベル 3	<ul style="list-style-type: none">• コネクターがビジネス・オブジェクトで外部キーを検出した時期または設定した時期、などの情報が格納される外部キー処理メッセージ。• ビジネス・オブジェクト処理についての情報を格納するメッセージ。例えば、これらのメッセージは、コネクターがビジネス・オブジェクト間の一致を検出したとき、または子ビジネス・オブジェクトの配列でビジネス・オブジェクトを検出したときにデリバリーされます。

表 14. PeopleSoft トレース・メッセージ (続き)

トレース・レベル	トレース・メッセージ
レベル 4	<ul style="list-style-type: none"> • アプリケーション固有情報メッセージ。例えば、ビジネス・オブジェクトのアプリケーション固有情報フィールドを構文解析する関数から戻された値を示すメッセージ。 • コネクタのプロセス・フローをトレースできる関数を、コネクタが開始または終了した時期を識別するメッセージ。 • すべてのスレッド固有のメッセージ。コネクタが複数のスレッドを作成する場合、各新規スレッドの作成時にメッセージが出されません。
レベル 5	<ul style="list-style-type: none"> • コネクタの初期設定を示すメッセージ。例えば、統合ブローカーから検索した各構成プロパティの値を示すメッセージ。 • アプリケーションで実行されるステートメントが格納されるメッセージ。このトレース・レベルでは、コネクタのログ・ファイルには、宛先アプリケーションで実行されるすべてのステートメントと置換される任意の変数の値が格納されます。 • コネクタがビジネス・オブジェクトを処理する前 (コネクタが受け取ったままの状態を表示) と処理を完了した後 (コネクタが統合ブローカーに戻す状態の表示) のビジネス・オブジェクトの表現で構成されるメッセージ。 • ビジネス・オブジェクト・ダンプを構成するメッセージ。 • コネクタが実行中に作成する各スレッドの状況を示すメッセージ。

エラー・メッセージ

コネクタが生成するすべてのエラー・メッセージは、PeopleSoftConnector.txt という名前のメッセージ・ファイルに格納されます。各エラーはエラー番号が付けられ、その後にエラー・メッセージが表示されます。例えば、次のようになります。

1210

PeopleSoft Connector unable to initialize.

1211

PeopleSoft Connector failed to locate.

アプリケーションへの接続の切断

コネクタのアプリケーション固有のコンポーネントが接続の確立に失敗すると、コネクタは統合ブローカーに FAIL を送信して終了します。

付録 A. コネクターの標準構成プロパティ

この付録では、WebSphere Business Integration Adapter のコネクター・コンポーネントの標準構成プロパティについて説明します。この付録の内容は、以下の統合ブローカーで実行されるコネクターを対象としています。

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator、WebSphere MQ Integrator Broker、および WebSphere Business Integration Message Broker (WebSphere Message Brokers (WMQI) と総称)
- WebSphere Application Server (WAS)

コネクターによっては、一部の標準プロパティが使用されないことがあります。Connector Configurator から統合ブローカーを選択するときには、そのブローカーで実行されるアダプターについて構成する必要のある標準プロパティのリストが表示されます。

コネクター固有のプロパティの詳細については、該当するアダプターのユーザーズ・ガイドを参照してください。

注: この資料では、ディレクトリー・パスに円記号 (¥) を使用します。UNIX システムを使用している場合は、円記号をスラッシュ (/) に置き換えてください。また、各オペレーティング・システムの規則に従ってください。

新規プロパティと削除されたプロパティ

以下の標準プロパティは、本リリースで追加されました。

新規プロパティ

- XMLNamespaceFormat

削除されたプロパティ

- RestartCount

標準コネクター・プロパティの構成

アダプター・コネクターには 2 つのタイプの構成プロパティがあります。

- 標準構成プロパティ
- コネクター固有の構成プロパティ

このセクションでは、標準構成プロパティについて説明します。コネクター固有の構成プロパティについては、該当するアダプターのユーザーズ・ガイドを参照してください。

Connector Configurator の使用

Connector Configurator からコネクタ・プロパティを構成します。Connector Configurator には、System Manager からアクセスします。Connector Configurator の使用法の詳細については、本書の Connector Configurator に関するセクションを参照してください。

注: Connector Configurator と System Manager は、Windows システム上でのみ動作します。コネクタを UNIX システム上で稼働している場合でも、これらのツールがインストールされた Windows マシンが必要です。UNIX 上で動作するコネクタのコネクタ・プロパティを設定する場合は、Windows マシン上で System Manager を起動し、UNIX の統合ブローカーに接続してから、コネクタ用の Connector Configurator を開く必要があります。

プロパティ値の設定と更新

プロパティ・フィールドのデフォルトの長さは 255 文字です。

コネクタは、以下の順序に従ってプロパティの値を決定します (最も番号の大きい項目が他の項目よりも優先されます)。

1. デフォルト
2. リポジトリ (WebSphere InterChange Server が統合ブローカーである場合のみ)
3. ローカル構成ファイル
4. コマンド行

コネクタは、始動時に構成値を取得します。実行時セッション中に 1 つ以上のコネクタ・プロパティの値を変更する場合は、プロパティの**更新メソッド**によって、変更を有効にする方法が決定されます。標準コネクタ・プロパティには、以下の 4 種類の更新メソッドがあります。

• 動的

変更を System Manager に保管すると、変更が即時に有効になります。コネクタが System Manager から独立してスタンドアロン・モードで稼働している場合 (例えば、いずれかの WebSphere Message Brokers と連携している場合) は、構成ファイルでのみプロパティを変更できます。この場合、動的更新は実行できません。

• エージェント再始動 (ICS のみ)

アプリケーション固有のコンポーネントを停止して再始動しなければ、変更が有効になりません。

• コンポーネント再始動

System Manager でコネクタを停止してから再始動しなければ、変更が有効になりません。アプリケーション固有コンポーネントまたは統合ブローカーを停止、再始動する必要はありません。

• サーバー再始動

アプリケーション固有のコンポーネントおよび統合ブローカーを停止して再始動しなければ、変更が有効になりません。

特定のプロパティの更新方法を確認するには、「Connector Configurator」ウィンドウ内の「更新メソッド」列を参照するか、次に示す107 ページの表 15 の「更新メソッド」列を参照してください。

標準プロパティの要約

表 15 は、標準コネクタ構成プロパティの早見表です。標準プロパティの依存関係は RepositoryDirectory に基づいているため、コネクタによっては使用されないプロパティがあり、使用する統合ブローカーによってプロパティの設定が異なる可能性があります。

コネクタを実行する前に、これらのプロパティの一部の値を設定する必要があります。各プロパティの詳細については、次のセクションを参照してください。

注: 表 15 の「注」列にある「Repository Directory は REMOTE」という句は、ブローカーが InterChange Server であることを示します。ブローカーが WMQI または WAS の場合には、リポジトリ・ディレクトリは LOCAL に設定されます。

表 15. 標準構成プロパティの要約

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
AdminInQueue	有効な JMS キュー名	CONNECTORNAME /ADMININQUEUE	コンポーネント再始動	Delivery Transport は JMS
AdminOutQueue	有効な JMS キュー名	CONNECTORNAME /ADMINOUTQUEUE	コンポーネント再始動	Delivery Transport は JMS
AgentConnections	1 から 4	1	コンポーネント再始動	Delivery Transport は MQ および IDL: Repository Directory は <REMOTE> (ブローカーは ICS)
AgentTraceLevel	0 から 5	0	動的	
ApplicationName	アプリケーション名	コネクタ・アプリケーション名として指定された値	コンポーネント再始動	
BrokerType	ICS、WMQI、WAS		コンポーネント再始動	
CharacterEncoding	ascii7、ascii8、SJIS、Cp949、GBK、Big5、Cp297、Cp273、Cp280、Cp284、Cp037、Cp437 注: これは、サポートされる値の一部です。	ascii7	コンポーネント再始動	
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント再始動	Repository Directory は <REMOTE> (ブローカーは ICS)
ContainerManagedEvents	値なしまたは JMS	値なし	コンポーネント再始動	Delivery Transport は JMS

表 15. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
ControllerStoreAndForwardMode	true または false	true	動的	Repository Directory は <REMOTE> (ブローカーは ICS)
ControllerTraceLevel	0 から 5	0	動的	Repository Directory は <REMOTE> (ブローカーは ICS)
DeliveryQueue		CONNECTORNAME/DELIVERYQUEUE	コンポーネント再始動	JMS トランスポートのみ
DeliveryTransport	MQ、IDL、または JMS	JMS	コンポーネント再始動	Repository Directory が ローカルの場合は、値は JMS のみ
DuplicateEventElimination	true または false	false	コンポーネント再始動	JMS トランスポートのみ: Container Managed Events は <NONE> でなければならぬ
FaultQueue		CONNECTORNAME/FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory または CxCommon.Messaging.jms.SonicMQFactory または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	FactoryClassName が IBM の場合は crossworlds.queue.manager を使用。FactoryClassName が Sonic の場合は localhost:2506 を使用。	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	Repository Directory は <REMOTE> (ブローカーは ICS)

表 15. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
JvmMaxNativeStackSize	スタックのサイズ (キロバイト単位)	128k	コンポーネント再始動	Repository Directory は <REMOTE> (ブローカーは ICS)
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	Repository Directory は <REMOTE> (ブローカーは ICS)
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならぬ
Locale	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR 注: これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	true または false	false	コンポーネント再始動	Repository Directory は <REMOTE> でなければならぬ (ブローカーは ICS)
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならぬ (ブローカーは ICS)
MessageFileName	パスまたはファイル名	CONNECTORNAMEConnector.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	CONNECTORNAME/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は true でなければならぬ
OADAutoRestartAgent	true または false	false	動的	Repository Directory は <REMOTE> でなければならぬ (ブローカーは ICS)
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければならぬ (ブローカーは ICS)

表 15. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければならぬ (ブローカーは ICS)
PollEndTime	HH:MM	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒) no (ポーリングを使用不可にする) key (コネクタのコマンド・プロンプト・ウィンドウで文字 p が入力された場合にのみポーリングする)	10000	動的	
PollQuantity	1 から 500	1	エージェント再始動	JMS トランスポートのみ: Container Managed Events を指定
PollStartTime	HH:MM (HH は 0 から 23、MM は 0 から 59)	HH:MM	コンポーネント再始動	
RepositoryDirectory	メタデータ・リポジトリの場所		エージェント再始動	ICS の場合は <REMOTE> に設定する。 WebSphere MQ Message Brokers および WAS の場合: C:¥crossworlds¥repository に設定する
RequestQueue	有効な JMS キュー名	CONNECTORNAME/REQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
ResponseQueue	有効な JMS キュー名	CONNECTORNAME/RESPONSEQUEUE	コンポーネント再始動	Delivery Transport が JMS の場合: Repository Directory が <REMOTE> の場合のみ必要
RestartRetryCount	0 から 99	3	動的	
RestartRetryInterval	適切な正数 (単位: 分): 1 から 2147483547	1	動的	
RHF2MessageDomain	mrm、xml	mrm	コンポーネント再始動	Delivery Transport が JMS であり、かつ WireFormat が CwXML である。

表 15. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
SourceQueue	有効な WebSphere MQ 名	CONNECTORNAME/SOURCEQUEUE	エージェント再始動	Delivery Transport が JMS であり、かつ Container Managed Events が指定されている場合のみ
SynchronousRequestQueue		CONNECTORNAME/ SYNCHRONOUSREQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
SynchronousRequestTimeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	Delivery Transport は JMS
SynchronousResponseQueue		CONNECTORNAME/ SYNCHRONOUSRESPONSEQUEUE	コンポーネント再始動	Delivery Transport は JMS
WireFormat	CwXML、CwBO	CwXML	エージェント再始動	Repository Directory が <REMOTE> でない場合は CwXML。Repository Directory が <REMOTE> であれば CwBO
WsifSynchronousRequestTimeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	WAS のみ
XMLNamespaceFormat	short、long	short	エージェント再始動	WebSphere MQ Message Brokers および WAS のみ

標準構成プロパティ

このセクションでは、各標準コネクタ構成プロパティの定義を示します。

AdminInQueue

統合ブローカーからコネクタへ管理メッセージが送信されるときに使用されるキューです。

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

コネクタから統合ブローカーへ管理メッセージが送信されるときに使用されるキューです。

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

RepositoryDirectory が <REMOTE> の場合のみ適用可能です。

AgentConnections プロパティは、orb.init[] により開かれる ORB (オブジェクト・リクエスト・ブローカー) 接続の数を制御します。

このプロパティのデフォルト値は 1 に設定されます。必要に応じてこの値を変更できます。

AgentTraceLevel

アプリケーション固有のコンポーネントのトレース・メッセージのレベルです。デフォルト値は 0 です。コネクタは、設定されたトレース・レベル以下の該当するトレース・メッセージをすべてデリバリーします。

ApplicationName

コネクタのアプリケーションを一意的に特定する名前です。この名前は、システム管理者が WebSphere Business Integration システム環境をモニターするために使用されます。コネクタを実行する前に、このプロパティに値を指定する必要があります。

BrokerType

使用する統合ブローカー・タイプを指定します。オプションは ICS、WebSphere Message Brokers (WMQI、WMQIB または WBIMB) または WAS です。

CharacterEncoding

文字 (アルファベットの文字、数値表現、句読記号など) から数値へのマッピングに使用する文字コード・セットを指定します。

注: Java ベースのコネクタでは、このプロパティは使用しません。C++ ベースのコネクタでは、現在、このプロパティに `ascii7` という値が使用されています。

デフォルトでは、ドロップダウン・リストには、サポートされる文字エンコードの一部のみが表示されます。ドロップダウン・リストに、サポートされる他の値を追加するには、製品ディレクトリーにある `¥Data¥Std¥stdConnProps.xml` ファイルを手動で変更する必要があります。詳細については、本書の Connector Configurator に関するセクションを参照してください。

ConcurrentEventTriggeredFlows

RepositoryDirectory が <REMOTE> の場合のみ適用可能です。

コネクタがイベントのデリバリー時に並行処理できるビジネス・オブジェクトの数を決定します。この属性の値を、並行してマップおよびデリバリーできるビジネス・オブジェクトの数に設定します。例えば、この属性の値を 5 に設定すると、5 個のビジネス・オブジェクトが並行して処理されます。デフォルト値は 1 です。

このプロパティを 1 よりも大きい値に設定すると、ソース・アプリケーションのコネクタが、複数のイベント・ビジネス・オブジェクトを同時にマップして、複

数のコラボレーション・インスタンスにそれらのビジネス・オブジェクトを同時にデリバリーすることができます。これにより、統合ブローカーへのビジネス・オブジェクトのデリバリーにかかる時間、特にビジネス・オブジェクトが複雑なマップを使用している場合のデリバリー時間が短縮されます。ビジネス・オブジェクトのコラボレーションに到達する速度を増大させると、システム全体のパフォーマンスを向上させることができます。

ソース・アプリケーションから宛先アプリケーションまでのフロー全体に並行処理を実装するには、次のようにする必要があります。

- `Maximum number of concurrent events` プロパティの値を増加して、コラボレーションが複数のスレッドを使用できるように構成します。
- 宛先アプリケーションのアプリケーション固有コンポーネントが複数の要求を並行して実行できることを確認します。つまり、このコンポーネントがマルチスレッド化されているか、またはコネクタ・エージェント並列処理を使用でき、複数プロセスに対応するよう構成されている必要があります。`Parallel Process Degree` 構成プロパティに、1 より大きい値を設定します。

`ConcurrentEventTriggeredFlows` プロパティは、順次に行われる単一スレッド処理であるコネクタのポーリングでは無効です。

ContainerManagedEvents

このプロパティにより、JMS イベント・ストアを使用する JMS 対応コネクタが、保証付きイベント・デリバリーを提供できるようになります。保証付きイベント・デリバリーでは、イベントはソース・キューから除去され、単一 JMS トランザクションとして宛先キューに配置されます。

デフォルト値はありません。

`ContainerManagedEvents` を JMS に設定した場合には、保証付きイベント・デリバリーを使用できるように次のプロパティも構成する必要があります。

- `PollQuantity` = 1 から 500
- `SourceQueue` = /SOURCEQUEUE

また、`MimeType`、`DHClass` (データ・ハンドラー・クラス)、および `DataHandlerConfigMOName` (オプションのメタオブジェクト名) プロパティを設定したデータ・ハンドラーも構成する必要があります。これらのプロパティの値を設定するには、`Connector Configurator` の「データ・ハンドラー」タブを使用します。

これらのプロパティはアダプター固有ですが、例の値は次のようになります。

- `MimeType` = `text/xml`
- `DHClass` = `com.crossworlds.DataHandlers.text.xml`
- `DataHandlerConfigMOName` = `MO_DataHandler_Default`

「データ・ハンドラー」タブのこれらの値のフィールドは、`ContainerManagedEvents` を JMS に設定した場合にのみ表示されます。

注: ContainerManagedEvents を JMS に設定した場合、コネクターはその pollForEvents() メソッドを呼び出さなくなるため、そのメソッドの機能は使用できなくなります。

このプロパティーは、DeliveryTransport プロパティーが値 JMS に設定されている場合にのみ表示されます。

ControllerStoreAndForwardMode

RepositoryDirectory が <REMOTE> の場合のみ適用可能です。

宛先側のアプリケーション固有のコンポーネントが使用不可であることをコネクター・コントローラーが検出した場合に、コネクター・コントローラーが実行する動作を設定します。

このプロパティーを true に設定した場合、イベントが ICS に到達したときに宛先側のアプリケーション固有のコンポーネントが使用不可であれば、コネクター・コントローラーはそのアプリケーション固有のコンポーネントへの要求をブロックします。アプリケーション固有のコンポーネントが作動可能になると、コネクター・コントローラーはアプリケーション固有のコンポーネントにその要求を転送します。

ただし、コネクター・コントローラーが宛先側のアプリケーション固有のコンポーネントにサービス呼び出し要求を転送した後でこのコンポーネントが使用不可になった場合、コネクター・コントローラーはその要求を失敗させます。

このプロパティーを false に設定した場合、コネクター・コントローラーは、宛先側のアプリケーション固有のコンポーネントが使用不可であることを検出すると、ただちにすべてのサービス呼び出し要求を失敗させます。

デフォルト値は true です。

ControllerTraceLevel

RepositoryDirectory が <REMOTE> の場合のみ適用可能です。

コネクター・コントローラーのトレース・メッセージのレベルです。デフォルト値は 0 です。

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

コネクターから統合ブローカーへビジネス・オブジェクトが送信されるときに使用されるキューです。

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

イベントのデリバリーのためのトランスポート機構を指定します。指定可能な値は、WebSphere MQ の MQ、CORBA IIOP の IDL、Java Messaging Service の JMS です。

- `RepositoryDirectory` がリモートの場合は、`DeliveryTransport` プロパティの指定可能な値は `MQ`、`IDL`、または `JMS` であり、デフォルトは `IDL` になります。
- `RepositoryDirectory` がローカル・ディレクトリーの場合は、指定可能な値は `JMS` のみです。

`DeliveryTransport` プロパティに指定されている値が、`MQ` または `IDL` である場合、コネクタは、`CORBA IIOP` を使用してサービス呼び出し要求と管理メッセージを送信します。

WebSphere MQ および IDL

イベントのデリバリー・トランスポートには、`IDL` ではなく `WebSphere MQ` を使用してください (1 種類の製品だけを使用する必要がある場合を除きます)。

`WebSphere MQ` が `IDL` よりも優れている点は以下のとおりです。

- 非同期 (ASYNC) 通信:
`WebSphere MQ` を使用すると、アプリケーション固有のコンポーネントは、サーバーが利用不能である場合でも、イベントをポーリングして永続的に格納することができます。
- サーバー・サイド・パフォーマンス:
`WebSphere MQ` を使用すると、サーバー・サイドのパフォーマンスが向上します。最適化モードでは、`WebSphere MQ` はイベントへのポインターのみをリポジトリ・データベースに格納するので、実際のイベントは `WebSphere MQ` キュー内に残ります。これにより、サイズが大きい可能性のあるイベントをリポジトリ・データベースに書き込む必要がありません。
- エージェント・サイド・パフォーマンス:
`WebSphere MQ` を使用すると、アプリケーション固有のコンポーネント側のパフォーマンスが向上します。`WebSphere MQ` を使用すると、コネクタのポーリング・スレッドは、イベントを選出した後、コネクタのキューにそのイベントを入れ、次のイベントを選出します。この方法は `IDL` よりも高速で、`IDL` の場合、コネクタのポーリング・スレッドは、イベントを選出した後、ネットワーク経由でサーバー・プロセスにアクセスしてそのイベントをリポジトリ・データベースに永続的に格納してから、次のイベントを選出する必要があります。

JMS

Java Messaging Service (JMS) を使用しての、コネクタとクライアント・コネクタ・フレームワークとの間の通信を可能にします。

`JMS` をデリバリー・トランスポートとして選択した場合は、

`jms.MessageBrokerName`、`jms.FactoryClassName`、`jms.Password`、`jms.UserName` などの追加の `JMS` プロパティが `Connector Configurator` 内に表示されます。このうち最初の 2 つは、このトランスポートの必須プロパティです。

重要: 以下の環境では、コネクタに `JMS` トランスポート機構を使用すると、メモリ制限が発生することもあります。

- AIX 5.0
- `WebSphere MQ` 5.3.0.1
- `ICS` が統合ブローカーの場合

この環境では、WebSphere MQ クライアント内でメモリーが使用されるため、(サーバー側の) コネクター・コントローラーと (クライアント側の) コネクターの両方を始動するのは困難な場合があります。ご使用のシステムのプロセス・ヒープ・サイズが 768M 未満である場合には、次のように設定することをお勧めします。

- CWSHaredEnv.sh スクリプト内で LDR_CNTRL 環境変数を設定する。

このスクリプトは、製品ディレクトリー配下の %bin ディレクトリーにあります。テキスト・エディターを使用して、CWSHaredEnv.sh スクリプトの最初の行として次の行を追加します。

```
export LDR_CNTRL=MAXDATA=0x30000000
```

この行は、ヒープ・メモリーの使用量を最大 768 MB (3 セグメント * 256 MB) に制限します。プロセス・メモリーがこの制限値を超えると、ページ・スワッピングが発生し、システムのパフォーマンスに悪影響を与える場合があります。

- IPCCBaseAddress プロパティーの値を 11 または 12 に設定する。このプロパティーの詳細については、「システム・インストール・ガイド (UNIX 版)」を参照してください。

DuplicateEventElimination

このプロパティーを true に設定すると、JMS 対応コネクターによるデリバリー・キューへの重複イベントのデリバリーが防止されます。この機能を使用するには、コネクターに対し、アプリケーション固有のコード内でビジネス・オブジェクトの **ObjectEventId** 属性として一意のイベント ID が設定されている必要があります。これはコネクター開発時に設定されます。

このプロパティーは、false に設定することもできます。

注: DuplicateEventElimination を true に設定する際は、MonitorQueue プロパティーを構成して保証付きイベント・デリバリーを使用可能にする必要があります。

FaultQueue

コネクターでメッセージを処理中にエラーが発生すると、コネクターは、そのメッセージを状況表示および問題説明とともにこのプロパティーに指定されているキューに移動します。

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

エージェントの最大ヒープ・サイズ (メガバイト単位)。このプロパティーは、RepositoryDirectory の値が <REMOTE> の場合にのみ適用されます。

デフォルト値は 128M です。

JvmMaxNativeStackSize

エージェントの最大ネイティブ・スタック・サイズ (キロバイト単位)。このプロパティーは、RepositoryDirectory の値が <REMOTE> の場合にのみ適用されます。

デフォルト値は 128K です。

JvmMinHeapSize

エージェントの最小ヒープ・サイズ (メガバイト単位)。このプロパティは、RepositoryDirectory の値が <REMOTE> の場合にのみ適用されます。

デフォルト値は 1M です。

jms.FactoryClassName

JMS プロバイダーのためにインスタンスを生成するクラス名を指定します。JMS をデリバリー・トランスポート機構 (DeliveryTransport) として選択する際は、このコネクタ・プロパティを必ず 設定してください。

デフォルト値は CxCommon.Messaging.jms.IBMMQSeriesFactory です。

jms.MessageBrokerName

JMS プロバイダーのために使用するブローカー名を指定します。JMS をデリバリー・トランスポート機構 (DeliveryTransport) として選択する際は、このコネクタ・プロパティを必ず 設定してください。

デフォルト値は crossworlds.queue.manager です。ローカル・メッセージ・ブローカーに接続する場合は、デフォルト値を使用します。

リモート・メッセージ・ブローカーに接続すると、このプロパティは次の (必須) 値をとります。

QueueMgrName:<Channel>:<HostName>:<PortNumber>

各変数の意味は以下のとおりです。

QueueMgrName: キュー・マネージャー名です。

Channel: クライアントが使用するチャンネルです。

HostName: キュー・マネージャーの配置先のマシン名です。

PortNumber: キュー・マネージャーが listen に使用するポートの番号です。

例えば、次のようになります。

```
jms.MessageBrokerName = WBIMB.Queue.Manager:CHANNEL1:RemoteMachine:1456
```

jms.NumConcurrentRequests

コネクタに対して同時に送信することができる並行サービス呼び出し要求の数 (最大値) を指定します。この最大値に達した場合、新規のサービス呼び出し要求はブロックされ、既存のいずれかの要求が完了した後で処理されます。

デフォルト値は 10 です。

jms.Password

JMS プロバイダーのためのパスワードを指定します。このプロパティの値はオプションです。

デフォルトはありません。

jms.UserName

JMS プロバイダーのためのユーザー名を指定します。このプロパティの値はオプションです。

デフォルトはありません。

ListenerConcurrency

このプロパティは、統合ブローカーとして ICS を使用する場合の MQ Listener でのマルチスレッド化をサポートしています。このプロパティにより、データベースへの複数イベントの書き込み操作をバッチ処理できるので、システム・パフォーマンスが向上します。デフォルト値は 1 です。

このプロパティは、MQ トランSPORTを使用するコネクタにのみ適用されます。DeliveryTransport プロパティには MQ を設定してください。

Locale

言語コード、国または地域、および、希望する場合には、関連した文字コード・セットを指定します。このプロパティの値は、データの照合やソート順、日付と時刻の形式、通貨記号などの国/地域別情報を決定します。

ロケール名は、次のフォーマットで指定します。

ll_TT.codeset

ここで、以下のように説明されます。

<i>ll</i>	2 文字の言語コード (普通は小文字)
<i>TT</i>	2 文字の国または地域コード (普通は大文字)
<i>codeset</i>	関連文字コード・セットの名前。名前のこの部分は、通常、オプションです。

デフォルトでは、ドロップダウン・リストには、サポートされるロケールの一部のみが表示されます。ドロップダウン・リストに、サポートされる他の値を追加するには、製品ディレクトリーにある `¥Data¥Std¥stdConnProps.xml` ファイルを手動で変更する必要があります。詳細については、本書の Connector Configurator に関するセクションを参照してください。

デフォルト値は `en_US` です。コネクタがグローバル化に対応していない場合、このプロパティの有効な値は `en_US` のみです。特定のコネクタがグローバル化に対応しているかどうかを判別するには、以下の Web サイトにあるコネクタのバージョン・リストを参照してください。

<http://www.ibm.com/software/websphere/wbiadapters/infocenter>、または
<http://www.ibm.com/websphere/integration/wicsserver/infocenter>

LogAtInterchangeEnd

RepositoryDirectory が <REMOTE> の場合のみ適用可能です。

統合ブローカーのログ宛先にエラーを記録するかどうかを指定します。ブローカーのログ宛先にログを記録すると、電子メール通知もオンになります。これにより、

エラーまたは致命的エラーが発生すると、InterchangeSystem.cfg ファイルに指定された MESSAGE_RECIPIENT に対する電子メール・メッセージが生成されます。

例えば、LogAtInterChangeEnd を true に設定した場合にコネクタからアプリケーションへの接続が失われると、指定されたメッセージ宛先に、電子メール・メッセージが送信されます。デフォルト値は false です。

MaxEventCapacity

コントローラー・バッファ内のイベントの最大数。このプロパティはフロー制御が使用し、RepositoryDirectory プロパティの値が <REMOTE> の場合にのみ適用されます。

値は 1 から 2147483647 の間の正整数です。デフォルト値は 2147483647 です。

MessageFileName

コネクタ・メッセージ・ファイルの名前です。メッセージ・ファイルの標準位置は、製品ディレクトリーの %connectors%messages です。メッセージ・ファイルが標準位置に格納されていない場合は、メッセージ・ファイル名を絶対パスで指定します。

コネクタ・メッセージ・ファイルが存在しない場合は、コネクタは InterchangeSystem.txt をメッセージ・ファイルとして使用します。このファイルは、製品ディレクトリーに格納されています。

注: 特定のコネクタについて、コネクタ独自のメッセージ・ファイルがあるかどうかを判別するには、該当するアダプターのユーザズ・ガイドを参照してください。

MonitorQueue

コネクタが重複イベントをモニターするために使用する論理キューです。このプロパティは、DeliveryTransport プロパティ値が JMS であり、かつ DuplicateEventElimination が TRUE に設定されている場合にのみ使用されます。

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

RepositoryDirectory が <REMOTE> の場合のみ有効です。

コネクタが自動再始動およびリモート再始動機能を使用するかどうかを指定します。この機能では、MQ により起動される Object Activation Daemon (OAD) を使用して、異常シャットダウン後にコネクタを再始動したり、System Monitor からリモート・コネクタを始動したりします。

自動再始動機能およびリモート再始動機能を使用可能にするには、このプロパティを true に設定する必要があります。MQ により起動される OAD 機能の構成方法については、「システム・インストール・ガイド (Windows 版)」または「システム・インストール・ガイド (UNIX 版)」を参照してください。

デフォルト値は false です。

OADMaxNumRetry

RepositoryDirectory が <REMOTE> の場合のみ有効です。

異常シャットダウンの後で MQ により起動される OAD がコネクタの再始動を自動的に試行する回数の最大数を指定します。このプロパティを有効にするためには、OADAutoRestartAgent プロパティを true に設定する必要があります。

デフォルト値は 1000 です。

OADRetryTimeInterval

RepositoryDirectory が <REMOTE> の場合のみ有効です。

MQ により起動される OAD の再試行時間間隔の分数を指定します。コネクタ・エージェントがこの再試行時間間隔内に再始動しない場合は、コネクタ・コントローラーはコネクタ・エージェントを再び再始動するように OAD に要求します。OAD はこの再試行プロセスを OADMaxNumRetry プロパティで指定された回数だけ繰り返します。このプロパティを有効にするためには、OADAutoRestartAgent プロパティを true に設定する必要があります。

デフォルト値は 10 です。

PollEndTime

イベント・キューのポーリングを停止する時刻です。形式は HH:MM です。ここで、HH は 0 から 23 時を表し、MM は 0 から 59 分を表します。

このプロパティには必ず有効な値を指定してください。デフォルト値は HH:MM ですが、この値は必ず変更する必要があります。

PollFrequency

これは、前回のポーリングの終了から次のポーリングの開始までの間の間隔です。PollFrequency は、あるポーリング・アクションの終了から次のポーリング・アクションの開始までの時間をミリ秒単位で指定します。これはポーリング・アクション間の間隔ではありません。この論理を次に説明します。

- ポーリングし、PollQuantity の値により指定される数のオブジェクトを取得します。
- これらのオブジェクトを処理します。一部のアダプターでは、これは個別のスレッドで部分的に実行されます。これにより、次のポーリング・アクションまで処理が非同期に実行されます。
- PollFrequency で指定された間隔にわたって遅延します。
- このサイクルを繰り返します。

PollFrequency は以下の値のいずれかに設定します。

- ポーリング・アクション間のミリ秒数 (整数)。
- ワード key。コネクタは、コネクタのコマンド・プロンプト・ウィンドウで文字 p が入力されたときのみポーリングを実行します。このワードは小文字で入力します。

- ワード no。コネクタはポーリングを実行しません。このワードは小文字で入力します。

デフォルト値は 10000 です。

重要: 一部のコネクタでは、このプロパティの使用が制限されています。このようなコネクタが存在する場合には、アダプターのインストールと構成に関する章で制約事項が説明されています。

PollQuantity

コネクタがアプリケーションからポーリングする項目の数を指定します。アダプターにコネクタ固有のポーリング数設定プロパティがある場合、標準プロパティの値は、このコネクタ固有のプロパティの設定値によりオーバーライドされます。

電子メール・メッセージもイベントと見なされます。コネクタは、電子メールに関するポーリングを受けたときには次のように動作します。

- コネクタは、1 回目のポーリングを受けると、メッセージの本文を選出します。これは、本文が添付とも見なされるからです。本文の MIME タイプにはデータ・ハンドラーが指定されていないので、コネクタは本文を無視します。
- コネクタは PO の最初の添付を処理します。この添付の MIME タイプには対応する DH があるので、コネクタはビジネス・オブジェクトを Visual Test Connector に送信します。
- 2 回目のポーリングを受けると、コネクタは PO の 2 番目の添付を処理します。この添付の MIME タイプには対応する DH があるので、コネクタはビジネス・オブジェクトを Visual Test Connector に送信します。
- これが受け入れられると、PO の 3 番目の添付が届きます。

PollStartTime

イベント・キューのポーリングを開始する時刻です。形式は HH:MM です。ここで、HH は 0 から 23 時を表し、MM は 0 から 59 分を表します。

このプロパティには必ず有効な値を指定してください。デフォルト値は HH:MM ですが、この値は必ず変更する必要があります。

RequestQueue

統合ブローカーが、ビジネス・オブジェクトをコネクタに送信するときに使用されるキューです。

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

コネクタが XML スキーマ文書を読み取るリポジトリの場所です。この XML スキーマ文書には、ビジネス・オブジェクト定義のメタデータが含まれています。

統合ブローカーが ICS の場合はこの値を <REMOTE> に設定する必要があります。これは、コネクタが InterChange Server リポジトリからこの情報を取得するためです。

統合ブローカーが WebSphere Message Broker または WAS の場合は、この値を <local directory> に設定する必要があります。

ResponseQueue

DeliveryTransport が JMS の場合のみ適用可能で、RepositoryDirectory が <REMOTE> の場合のみ必須です。

JMS 応答キューを指定します。JMS 応答キューは、応答メッセージをコネクタ・フレームワークから統合ブローカーへデリバリーします。統合ブローカーが ICS の場合、サーバーは要求を送信し、JMS 応答キューの応答メッセージを待ちます。

RestartRetryCount

コネクタによるコネクタ自体の再始動の試行回数を指定します。このプロパティを並列コネクタに対して使用する場合、コネクタのマスター側のアプリケーション固有のコンポーネントがスレーブ側のアプリケーション固有のコンポーネントの再始動を試行する回数が指定されます。

デフォルト値は 3 です。

RestartRetryInterval

コネクタによるコネクタ自体の再始動の試行間隔を分単位で指定します。このプロパティを並列コネクタに対して使用する場合、コネクタのマスター側のアプリケーション固有のコンポーネントがスレーブ側のアプリケーション固有のコンポーネントの再始動を試行する間隔が指定されます。指定可能な値の範囲は 1 から 2147483647 です。

デフォルト値は 1 です。

RHF2MessageDomain

WebSphere Message Brokers および WAS でのみ使用されます。

このプロパティにより、JMS ヘッダーのドメイン名フィールドの値を構成できます。JMS トランスポートを介してデータを WMQI に送信するときに、アダプター・フレームワークにより JMS ヘッダー情報、ドメイン名、および固定値 mrm が書き込まれます。この構成可能なドメイン名により、ユーザーは WMQI ブローカーによるメッセージ・データの処理方法を追跡できます。

サンプル・ヘッダーを以下に示します。

```
<mcd><Msd>mrm</Msd><Set>3</Set><Type>
Retek_POPhyDesc</Type><Fmt>CwXML</Fmt></mcd>
```

デフォルト値は mrm ですが、このプロパティには xml も設定できます。このプロパティは、DeliveryTransport が JMS に設定されており、かつ WireFormat が CwXML に設定されている場合にのみ表示されます。

SourceQueue

DeliveryTransport が JMS で、ContainerManagedEvents が指定されている場合のみ適用されます。

JMS イベント・ストアを使用する JMS 対応コネクタでの保証付きイベント・デリバリーをサポートするコネクタ・フレームワークに、JMS ソース・キューを指定します。詳細については、113 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

同期応答を要求する要求メッセージを、コネクタ・フレームワークからブローカーにデリバリーします。このキューは、コネクタが同期実行を使用する場合にのみ必要です。同期実行の場合、コネクタ・フレームワークは、SynchronousRequestQueue にメッセージを送信し、SynchronousResponseQueue でブローカーから戻される応答を待機します。コネクタに送信される応答メッセージには、元のメッセージの ID を指定する相関 ID が含まれています。

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

同期要求に対する応答として送信される応答メッセージを、ブローカーからコネクタ・フレームワークにデリバリーします。このキューは、コネクタが同期実行を使用する場合にのみ必要です。

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

コネクタが同期要求への応答を待機する時間を分単位で指定します。コネクタは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージをエラー・メッセージとともに障害キューに移動します。

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

- RepositoryDirectory がローカル・ディレクトリーの場合は、設定は CwXML になります。
- RepositoryDirectory の値が <REMOTE> の場合には、設定値は CwBO です。

WsifSynchronousRequestTimeout

WAS 統合ブローカーでのみ使用されます。

コネクタが同期要求への応答を待機する時間を分単位で指定します。コネクタは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージをエラー・メッセージとともに障害キューに移動します。

デフォルト値は 0 です。

XMLNamespaceFormat

WebSphere Message Brokers および WAS 統合ブローカーでのみ使用されます。

ビジネス・オブジェクト定義の XML 形式でネーム・スペースを short と long のどちらにするかをユーザーが指定できるようにするための、強力なプロパティです。

デフォルト値は short です。

付録 B. コネクタ固有のプロパティ

この付録では、実行時にこのコネクタに固有の構成プロパティについて説明します。コネクタ固有のプロパティには、コネクタ内の静的情報やロジックを、コード変更したり再ビルドしたりする必要なしに、変更する手段が用意されています。

表 16 に、コネクタのコネクタ固有構成プロパティを示します。プロパティの説明については、以下の各セクションを参照してください。

表 16. コネクタ固有の構成プロパティ

名前	指定可能な値	デフォルト	
		値	必要
AppServerMachineNameOrIP		n/a	はい
ApplicationPassword	コネクタのユーザー・アカウントのパスワード	PS	はい
ApplicationUserName	コネクタのユーザー・アカウントの名前	PS	はい
ConnectorID	イベント分散用のコネクタ名	null	いいえ
ConvertToPrimitiveFloat	true または false	false	いいえ
DisableCrossReferencing	true または false	false	いいえ
EventKeyDelimiter	<NameValueChar><DelimiterChar> 注: 実際の文字の間にスペースを指定しないでください。	=:	いいえ
PInGCompInterface	< CHANGEME > を、既存のコンポーネント・インターフェース名に置き換えます。	CHANG ME	はい
PollQuantity	1 から 500 までの値	1	いいえ
PortNumber	Jolt ポート番号	9000	はい
ReconnectSessionOnGetFail	true または false	false	いいえ
SetLangCode	1 以上の値	1	いいえ
SetLangCode	true または false	false	いいえ
PollFutureEvents	true または false	false	いいえ

AppServerMachineNameOrIP

アプリケーション・サーバーを実行しているマシンの名前または IP アドレス。コネクタは、アプリケーションに接続する場合にのみこの値を使用します。

ApplicationPassword

PeopleSoft アプリケーションのコネクタのユーザー・アカウントのパスワード。

デフォルト値はありません。

ApplicationUserName

PeopleSoft アプリケーションのコネクタのユーザー・アカウントの名前。

デフォルト値はありません。

ConnectorID

このプロパティの値は、`cweventci` を使用して `CW_EVENT_TBL` 内のイベント検索するのに使用されます。特定のイベントを検索するために複数のコネクタが使用される場合に、イベント分散のために使用されます。この値がブランクまたは `null` の場合、このプロパティは、イベント・コンポーネント・インターフェースで `Find()` メソッドを呼び出す前に設定されません。

ConvertToPrimitiveFloat

浮動オブジェクトをプリミティブ浮動オブジェクトに変換するかどうかを指定します。

デフォルト値は `false` です。

DisableCrossReferencing

キー値の設定方法を指定します。PeopleCode を介してキー値を生成する場合は、このプロパティを `false` に設定します。これにより、相互参照が使用可能になります。キー値をビジネス・オブジェクト定義のキー属性と対立する値に設定した場合は、このプロパティを `true` に設定します。

デフォルト値は `false` です。

EventKeyDelimiter

名前と値を区切る文字 (`NameValue` 文字) と、対をお互いに区切る文字 (`Delimiter` 文字) の名前と値の対で 2 文字を指定します。次の例は、デフォルトの等号 (=) を `NameValue` 文字に、デフォルトのコロン (:) を `Delimiter` 文字に使用します。

```
SETID=1234:DEPTID=5678
```

デフォルト値は `=:` です。

PingComplInterface

ネットワーク接続/セッションの妥当性をテストするため、コンポーネント・インターフェースに対し `ping` を実行します。セッションまたは接続が無効な場合には `FAIL` が戻されます。コンポーネント・インターフェース名が指定されている場合に、無効なセッションまたはネットワーク接続の問題が原因でセッション・インスタンスがこのインターフェースを取得できないと、コネクタ・エージェントが終了します。

PollQuantity

コネクタがポーリング間隔ごとに検索するデータベース表の行数。許容値は 1 から 500 です。

デフォルトは、1 です。

PortNumber

アプリケーション・サーバー上の Jolt ポート番号 (Tuxedo ポート番号ではありません)。コネクタは JSL に接続します。WSL ではありません。

デフォルト値は 9000 です。

ReconnectSessionOnGetFail

true に設定すると、コネクタは自動的に新しいセッション・オブジェクトを作成して、PeopleSoft アプリケーションに再接続します。コネクタは、コンポーネント・インターフェースの Get() メソッドがコネクタのセッション・オブジェクトを終了させる重要でないエラーまたは警告メッセージを戻す場合にのみ、このプロパティを使用します。通常、この問題はインスタンスのキーがコネクタが使用するキーと異なる場合に発生します。

デフォルト値は false です。

SessionPoolSize

コネクタで複数の PeopleSoft セッション・インスタンスを実行できます。このプロパティには、実行するセッション・インスタンスの数を設定します。インバウンド・オブジェクトに対してのみマルチスレッド化が可能です。フリー・セッションのプールから各インバウンド・オブジェクト・スレッドにセッション・インスタンスが割り振られ、そのセッション・インスタンスはビジー・プールに移動します。トランザクションが終了すると、セッション・インスタンスは自由プールにリリースされます。

コネクタをマルチスレッドで実行している場合、単一スレッドの順次実行ポーリング用に 1 つ、セッション・インスタンスが予約されます。

SetLangCode

コネクタが PeopleSoft アプリケーションに接続後、ただちに基本言語を設定するかどうかを指定します。true に設定すると、コネクタは回避策を使用して、言語表ではなく基本表の基本言語を正しく更新します。このプロパティを false に設定すると、基本表が正しく更新されないことがあります。

デフォルト値は false です。

PollFutureEvents

将来の日付のイベントにおいてポーリングの準備ができていないか確認するために、コネクタがこれらのイベントを検索するかどうかを指定します。これが true に設定されている場合、コネクタは最初に (将来のポーリング用に設定されていることを示す) 状況 99 のイベントを検索することにより、ポーリングを行います。このようなイベントを検出した場合、コネクタはシステム日付と照合してこれらのイベントの日付を検査します。イベント日付がシステム日付と同じかこれより小さい場合、コネクタはイベント状況を 0 に変更します。これは、このイベントにおいてポーリングの準備ができていないことを示します。

この値が `false` に設定されている場合、コネクタは将来のポーリングが設定されているイベントの有無を検査しません。

デフォルト値は `false` です。

付録 C. 特記事項

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態を提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Burlingame Laboratory Director
IBM Burlingame Laboratory
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができませんが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性があります。その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確認できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者をお願いします。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があります、単に目標を示しているものです。

著作権使用許諾

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

プログラミング・インターフェース情報

プログラミング・インターフェース情報は、プログラムを使用してアプリケーション・ソフトウェアを作成する際に役立ちます。

一般使用プログラミング・インターフェースにより、お客様はこのプログラム・ツール・サービスを含むアプリケーション・ソフトウェアを書くことができます。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

警告: 診断、修正、調整情報は、変更される場合がありますので、プログラミング・インターフェースとしては使用しないでください。

商標

以下は、IBM Corporation の商標です。

IBM
IBM ロゴ
AIX
CrossWorlds
DB2
DB2 Universal Database
Lotus
Lotus Domino
Lotus Notes
MQIntegrator
MQSeries
Tivoli
WebSphere

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

MMX、Pentium および ProShare は、Intel Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。



IBM WebSphere Business Integration Adapter Framework V2.4.0



Printed in Japan