WebSphere Business Integration Server
Express and Express Plus

Adapter for PeopleSoft ユーザーズ・ガイド

バージョン 4.3

WebSphere Business Integration Server
Express and Express Plus

Adapter for PeopleSoft ユーザーズ・ガイド

バージョン 4.3

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、 115 ページの『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、IBM WebSphere Business Integration Server Express バージョン 4.3、IBM WebSphere Business Integration Server Express Plus バージョン 4.3、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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： WebSphere Business Integration Server Express and Express Plus Adapter for PeopleSoft User Guide Version 4.3

発　行： 日本アイ・ビー・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 2004.8

目次

本書について

製品 IBM(R) WebSphere(R) Business Integration Server Express および IBM(R) WebSphere(R) Business Integration Server Express Plus は、InterChange Server Express、関連する Toolset Express、CollaborationFoundation、およびソフトウェア統合アダプターのセットで構成されています。Toolset に含まれるツールは、ビジネス・プロセスの作成、変更、および管理に役立ちます。プリパッケージされている各種アダプターは、お客様の複数アプリケーションにまたがるビジネス・プロセスに応じて、いずれかを選べるようになっています。標準的な処理のテンプレートである CollaborationFoundation は、カスタマイズされたプロセスを簡単に作成できるようにするためのものです。

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 Server Express Plus アダプターの開発について、理解している必要があります。

関連文書

本書の対象製品の一連の関連文書には、WebSphere Business Integration Server Express Plus のどのインストールにも共通する機能とコンポーネントの解説のほか、特定のコンポーネントに関する参考資料が含まれています。

関連文書は、http://www.ibm.com/websphere/wbiserverexpress/infocenter でダウンロード、インストール、および表示することができます。

注: 本書の発行後に公開されたテクニカル・サポートの技術情報や速報に、本書の対象製品に関する重要な情報が記載されている場合があります。これらの技術情報や速報は、WebSphere Business Integration のサポート Web サイト (http://www.ibm.com/software/integration/websphere/support/) で参照できます。適切なコンポーネント領域を選択し、「Technotes (技術情報)」セクションと「Flashes (速報)」セクションを参照してください。

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*	変数名または相互参照を示します。
青い文字	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。

命名規則

本書では、次の命名規則を適用します。

- Adapter for PeopleSoft のコネクター・コンポーネントは、単にコネクターと呼びます。
- 「コネクター」とは、Vision コネクター・フレームワークとコネクター・モジュールの組み合わせを指します。

本リリースの新機能

本書の最初のリリースです。

第 1 章 アダプターの概要

この章では、IBM WebSphere Business Integration Server Express Plus の Adapter for PeopleSoft 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 InterChange Server Express が PeopleTools バージョン 8.18 以降 (8.40、8.41、8.42、8.43) を使用する PeopleSoft バージョン 8 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、InterChange Server Express 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、InterChange Server Express 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

本書には、コネクター・フレームワークとアプリケーション固有のコンポーネント (本書でコネクターと呼ばれているもの) に関する情報が含まれています。次のセクションが含まれています。

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

InterChange Server Express とコネクターの関係の詳細については、「*システム管理ガイド*」を参照してください。

コネクター・コンポーネント

Adapter for PeopleSoft には、コネクターと、少なくとも 1 つの 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組み込まれています。図 1 に、コネクターと、PeopleSoft アプリケーションとの関連を示します。統合ブローカーは InterChange Server Express です。

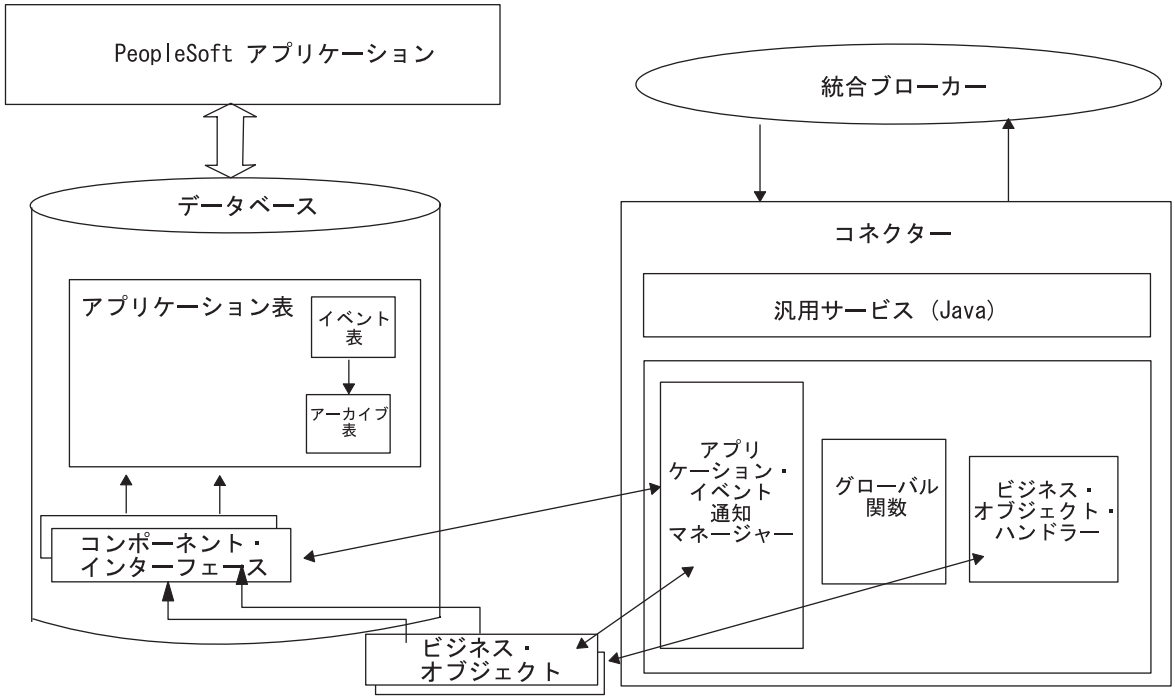

図 1. コネクター・アーキテクチャー

Connector for PeopleSoft 8

Java で作成されたコネクターは、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。コネクターは PeopleSoft セッション・オブジェクトを確立し、標準コネクター・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

コネクターは、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、コネクターを実装するユーザーが作成する必要があります。ただし、ビジネス・オブジェクトの作成を支援するために、アダプターにはサンプルの PeopleSoft 固有のビジネス・オブジェクトが用意されています。このサンプルは、製品ディレクトリー内の `connectors¥PeopleSoft¥samples` ディレクトリーに格納されています。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、33 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、38 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。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ー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、7 ページの『アプリケーション・イベントの処理』、21 ページの『cw_publish_events() 関数』、および 22 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ー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。

- メニュー定義

  CW_EVENT_MNU メニュー定義により、イベント・ページをオンラインで表示できます。

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。

- 『PeopleSoft アプリケーションとの対話』
- 4 ページの『ビジネス・オブジェクト要求の処理』
- 7 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 InterChange Server Express 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、31 ページの『第 3 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、InterChange Server Express 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ーに格納されます。
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、コネクターは `set`*`FieldName`*`(`*`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、コネクターは、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、54 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 PSMessage 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、コネクターは get*FieldName*() 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ーが処理する方法の詳細については、31 ページの『第 3 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。コネクター固有のプロパティーの詳細については、24 ページの『コネクター固有のプロパティー』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、コネクターはコンポーネント・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、コネクターは標準の 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. コネクターは、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。

コネクターがビジネス・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、コネクター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、コネクターは終了します。

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。

作成要求の処理

InterChange Server Express 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、コネクターは 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、43 ページの『作成操作』を参照してください。

検索要求の処理

InterChange Server Express 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、コネクターは 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、コネクター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。コネクターが InterChange Server Express 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、InterChange Server Express 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ー・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、44 ページの『検索操作』を参照してください。

更新要求の処理

InterChange Server Express 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、コネクターはコンポーネント・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。コネクターによりすべての単純なプロパティー・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。コネクターは、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、47 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、コネクター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、コネクター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。

存在要求の処理

InterChange Server Express が Exists 動詞を使用して階層ビジネス・オブジェクト要求を送信すると、コネクターはコンポーネント・インターフェースのインスタンスを検査します。コネクターのアプリケーション固有コンポーネントから、アプリケーションにインスタンスが存在する場合は `SUCCEED` が戻され、オブジェクトが存在しない場合は `FAIL` が戻されます。

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ーのイベント表 (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。
- 8 ページの『イベントのポーリング』: コネクターは PeopleSoft API を使用してイベント表をポーリングします。
- 8 ページの『イベントのアーカイブ』: コネクター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ー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、3 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_events()` 関数を使用します。この関数は FUNCLIB_CW レコードの FieldFormula イベントに格納されています。この関数は、イベントに関係するコンポーネントの `SavePostChg()` PeopleCode で宣言し、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、21 ページの『cw_publish_events() 関数』を参照してください。

イベントのポーリング

コネクターは、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、次にコネクター・プロパティー 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、プロパティーは検索に使用されません。コネクター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。コネクターの初期設定時は、保留中の INRPOGRESS イベントは READYFORPOLL (値 0) にリセットされます。

コネクターは 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。コネクター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。

コネクターは API を使用してプロパティー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 InterChange Server Express 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、コネクター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。コネクターは 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、コネクターはそれを InterChange Server Express 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、コネクター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。
- このコネクターは、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 InterChange Server Express に送信します。

詳細については、22 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。

イベントのアーカイブ

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、コネクター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

図 2 に、イベント通知プロセス・フローを示します。統合ブローカーは InterChange Server Express です。

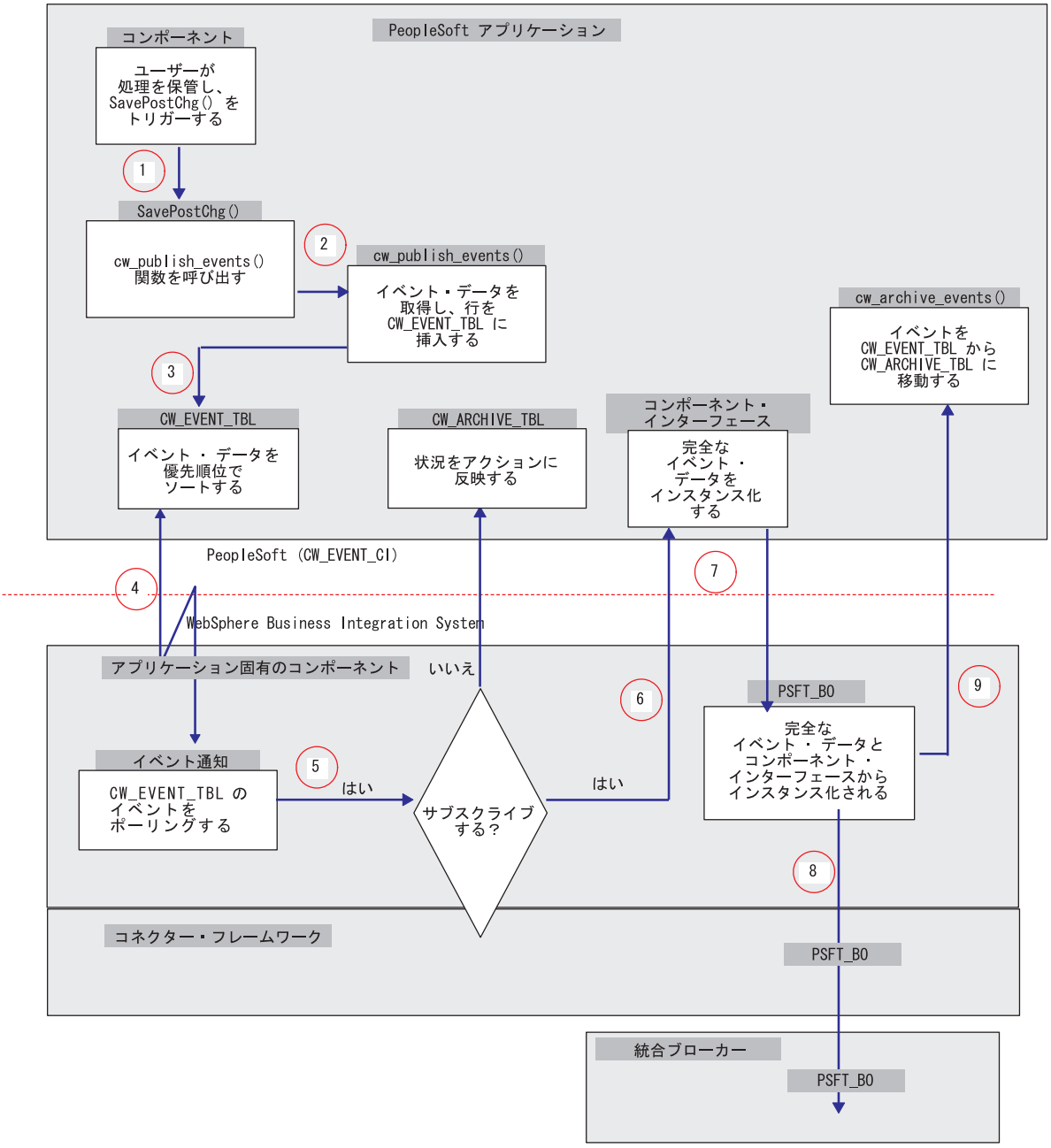

図 2. イベント通知プロセス・フロー

次に、図 2 に示すステップを説明します。

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() メソッドが cw_publish_events() 関数を呼び出します。
2. 4 つのパラメーターを使用する cw_publish_events() 関数で、次が実行されます。
   - 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。

- 3 番目のパラメーターの値に基づいて、イベントの優先順位を設定します。

  注: アプリケーションはコンポーネントでの 1 つの保管アクションで複数のイベントをパブリッシュできるため、コネクターは 3 番目のパラメーターの値を使用してイベントを優先順位付けします。オブジェクトにとって順序が重要である場合は、`cw_publish_events()` 関数を呼び出す際に優先順位を指定します。例えば、ロケーション (Location) は最初に処理する必要があるので、ロケーションは部門 (Department) より高い優先順位となります。

- `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` コネクター・プロパティーにも対応します。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。

詳細については、21 ページの『cw_publish_events() 関数』を参照してください。

3. コネクターは、CW_EVENT_TBL 内のイベントを優先順位でソートします。
4. コネクターは、その PollFrequency 構成プロパティー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 構成プロパティーで指定された数を超えません。ポーリングを行うために、コネクターは 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: コネクターは、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。

5. イベント表から取り出された各レコードについて、コネクター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、コネクターはイベント・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。

6. データがサブスクライブされる場合、コネクターはイベント・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。
7. コネクターは、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。
8. コネクターは、アプリケーション固有のビジネス・オブジェクトを InterChange Server Express に送信します。
9. コネクターは `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。

第 2 章 コネクターのインストールと構成

この章では、Adapter for PeopleSoft 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ー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。

互換性

Adapter for PeopleSoft ガイドは、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 Server Express Plus Adapter Framework
- **統合ブローカー:** InterChange Server Express

注:

InterChange Server Express のインストール手順およびその前提条件については、「*WebSphere Business Integration Server Express インストール・ガイド*」を参照してください。

前提条件

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。

コネクターを使用するために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 ポートを経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。

  Windows

  ```
  %PS_HOME%¥web¥PSJOA
  ```

  Windows の終り

  このファイルを 16 ページの『Windows インストール済みファイルの構造』に説明されている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、psjoa.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 (38 ページの『API の生成』を参照)。

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のある、必須操作について説明します。

- CLASSPATH 環境変数を次のディレクトリーにマップします。

  Windows

  ```
  %ProductDir%¥connectors¥PeopleSoft¥dependencies
  ```

  Windows の終り

  psjoa.jar が CLASSPATH に定義されていることも確認します。
- 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、 15 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、15 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。
- 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、38 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

コネクター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ーの場合、コネクターのユーザー・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

コネクターはコンポーネント・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・プロセッサー) 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・プロセッサーが終了し、コネクターは 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、コネクターのユーザー・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

アダプターのインストールについては、次のサイトの WebSphere Business Integration Server Express InfoCenter にある 「*WebSphere Business Integration Server Express インストール・ガイド*」の、Adapter Capacity Pack for WebSphere Business Integration Server Express Plus からのインストールについての説明を参照してください。

http://www.ibm.com/websphere/wbiserverexpress/infocenter

インストール済みファイルの構造

Windows インストール済みファイルの構造

表 1 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 1. 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

%*ProductDir*S% のサブディレクトリー	**説明**
¥connectors¥PeopleSoft	• コネクターの CWPeopleSoft.jar と start_PeopleSoft.bat ファイルが格納されます。 • 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。 • SavePostChg.txt という名前のファイルが格納されます。このファイルにはイベントの処理に必要な PeopleCode のサンプルが格納されています。
..¥bin	このフォルダーに CWConnEnv.bat ファイルが含まれていることを確認してください。
..¥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	• 現行バージョンの PeopleTools からコピーする必要のある 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	PeopleSoftConnector.txt ファイルと、PeopleSoftConnector*_ll_TT*.txt ファイル (言語 (*_ll*) および国/地域 (*_TT*) に固有のメッセージ・ファイル) が格納されます。
¥connectors¥PeopleSoft¥samples	サンプルの PeopleSoft 固有のビジネス・オブジェクトが格納されます。
¥repository¥PeopleSoft¥	CN_PeopleSoft.txt ファイルが格納されます。

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。

- 使用する PeopleSoft アプリケーションに、必要なコンポーネント、コンポーネント・インターフェース、メニュー、ページ、レコード、および PeopleCode を定義します。必要なビジネス・オブジェクトの定義プロセスを簡素化するために、アダプターにはサンプルのビジネス・オブジェクトが用意されています。詳細については、31 ページの『第 3 章 コネクター用のビジネス・オブジェクトについて』および 57 ページの『第 4 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。
- 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。

- 『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』
- 21 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、コネクター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のある操作について説明します。コネクター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』
- 18 ページの『必要なオブジェクトのビルド』
- 19 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。
   *%ProductDir*S%¥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。プロジェクト名は CW_EVENT_*Vx* です。ここで、*Vx* はバージョン番号を示します。
2. PeopleSoft のアプリケーション・デザイナーを開き、「ファイル」メニューの「Copy Project From File...」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 3 に、PeopleSoft 内の画面を示します。

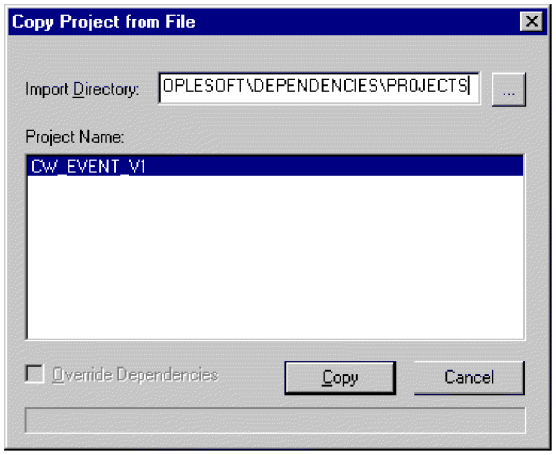

図 3. プロジェクトのインポート

4. 「コピー」をクリックして、3 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。

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。

1. 「Build」メニューの「Project...」を選択して、表をビルドする。
2. 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 4 のように表示されることを確認する。

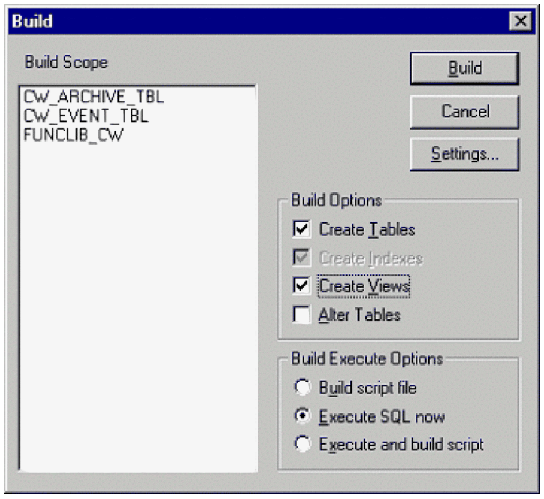

図 4. 表のビルド

3. 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 (コネクターがイベントのみを処理する場合に必要) またはコンポーネント・インターフェース API ファイル (コネクターがイベントと要求の両方を処理する場合に必要) 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ーの dependencies ディレクトリーへのパス (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ー) を入力する。通常は、次のとおりです。

Windows

```
%ProductDir%¥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 5 に、PeopleSoft 内の画面を示します。

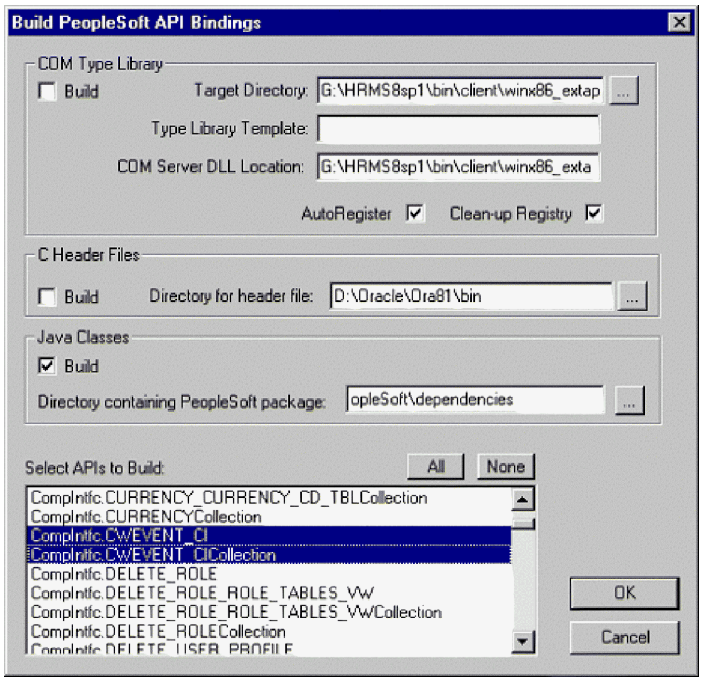

図 5. API のビルド

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、21 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、21 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

アダプターは、FUNCLIB_CW 関数ライブラリーに cw_publish_events() 関数を用意しています。この関数により、イベントがコネクターのイベント表に挿入されます。このセクションでは、これを呼び出す関数とサンプル・コードについて説明します。

cw_publish_events() 関数

cw_publish_events() 関数は、4 つのパラメーターを使用します。その型はすべて String です。

- &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。
- &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、26 ページの『EventKeyDelimiter』を参照してください。
- &CWPRIORITY1: 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。CW_EVENT_TBL により、優先順位のより高いイベントが最初に処理されます。
- &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 プロパティーに定義されます。イベント分散を使用しない場合は、値「PeopleSoftConnector」を使用します。

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

アダプターには、コネクターの samples ディレクトリーの savepostchg.txt ファイルに、PeopleCode 宣言と関数呼び出しのサンプルが格納されています。ビジネ

ス・オブジェクトの名前とキーを、使用するビジネス・オブジェクトの正しい情報で置き換えて、コードを PeopleCode エディターに直接コピーおよび貼り付けます。

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、コネクター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、コネクター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。

次のコードは、アダプターに用意されているサンプルです。

```
/* 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 String &CWPRIORITY1;
Component String &CONNID1;

   &BONAME1 ="Psft_Dept";
   &KEYLIST1 = "DEPT_TBL.SetId:DEPT_TBL.DeptId";
   &CWPRIORITY = 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,&CWPRIORITY1,&CONNID1);

End-if;
```

イベントおよびアーカイブ表

コネクター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、コネクター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ー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ーが変更されると、コネクター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、コネクターがイベント表からエンティティーを取得する前にエンティティーが変更および更新されると、コネクターは 1 回の取り出しで両方のデータ変更を取得します。

コネクター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。

コネクター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 2 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 2. 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ーの固有な ID。この値は、複数のコネクターが同じ表をポーリングする場合には重要です。	VARCHAR	
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、26 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	イベント優先順位 (0 が最高、n が最低) で、コネクターが優先順位に基づいてイベントを取得するために使用します。	NUMBER	非ヌル
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 (InterChange Server Express 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 (InterChange Server Express) 2 (ビジネス・オブジェクトのサブスクリプションなし) 3 (処理中)。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。	VARCHAR	非ヌル

コネクターの構成

コネクターの標準およびコネクター固有のコネクター構成プロパティーを構成してから、コネクターを実行する必要があります。コネクターのプロパティーを構成するには、次のものを使用します。

- Connector Designer (統合ブローカーが InterChange Server Express の場合。)

  System Manager からこのツールにアクセスします。

構成値を入力すると、値はリポジトリーに保管されます。

標準コネクター・プロパティー

標準の構成プロパティーにより、すべてのコネクターによって使用される情報が提供されます。これらのプロパティーの詳細については、 83 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

表 3 に、付録にある構成プロパティーについての、このコネクターに固有の情報を示します。

表 3. このコネクターに固有のプロパティー情報

プロパティー	注
AgentConnections	このコネクターは単一スレッドなので、このプロパティーのデフォルト値を変更しないでください。
CharacterEncoding	このコネクターはこのプロパティーを使用しません。
Locale	このコネクターは国際化対応されているため、このプロパティーの値は変更することができます。現在サポートされているロケールを調べるには、アダプターのリリース情報を参照してください。

コネクター固有のプロパティー

このセクションでは、実行時にこのコネクターに固有の構成プロパティーについて説明します。コネクター固有のプロパティーには、コネクター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 4 に、コネクターのコネクター固有構成プロパティーを示します。プロパティーの説明については、以下の各セクションを参照してください。

表 4. コネクター固有の構成プロパティー

名前	指定可能な値	デフォルト値	必要
AppServerMachineNameOrIP		n/a	はい
ApplicationPassword	コネクターのユーザー・アカウントのパスワード	PS	はい
ApplicationUserName	コネクターのユーザー・アカウントの名前	PS	はい
ConnectorID	イベント分散用のコネクター名	null	いいえ
ConvertToPrimitiveFloat	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。	CHANGME	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
	1 以上の値	1	いいえ
SetLangCode	true または false	false	いいえ
UseDefaults	true または false	false	はい
Version	PeopleTools バージョン番号	8.15	はい

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。コネクターは、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ーのユーザー・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ーのユーザー・アカウントの名前。

デフォルト値はありません。

ConnectorID

このプロパティーの値は、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ー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、このプロパティーは、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。

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。

PIngCompInterface

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、コネクター・エージェントが終了します。

PollQuantity

コネクター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)。コネクターは 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、コネクター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。コネクターは、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ーのセッション・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、このプロパティーを使用します。通常、この問題はインスタンスのキーがコネクター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

コネクターで複数の PeopleSoft セッション・インスタンスを実行できます。このプロパティーには、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。

コネクター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

コネクターが 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、コネクター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。このプロパティーを 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。

UseDefaults

UseDefaults が true に設定されているかまたは設定されていない場合、コネクターは必要な各ビジネス・オブジェクト属性について有効な値やデフォルト値が設定されているかどうかを検査します。値が設定されていると、Create は成功します。設定されていない場合は、失敗します。

UseDefaults が false に設定されている場合、コネクターは必要な各ビジネス・オブジェクト属性について有効な値が設定されているかどうかのみを検査します。有効な値が設定されていないと、Create 操作は失敗します。

デフォルト値は false です。

Version

アプリケーションが実行する PeopleTools の現行バージョンを指定します。値は、使用可能な最新の番号を 10 進小数点で指定します。

複数のコネクター・インスタンスの作成

コネクターの複数のインスタンスを作成する作業は、いろいろな意味で、カスタム・コネクターの作成と同じです。

注: WebSphere Business Integration Server Express に付属するアダプターの追加インスタンスは、配置できるアダプターの総数を制限する機能では、それぞれ別個のアダプターとして取り扱われます。

以下に示すステップを実行することによって、コネクターの複数のインスタンスを作成して実行するように、ご使用のシステムを設定することができます。次のようにする必要があります。

- コネクター・インスタンス用に新規ディレクトリーを作成します。
- 必要なビジネス・オブジェクト定義が設定されていることを確認します。
- 新規コネクター定義ファイルを作成します。
- 新規始動スクリプトを作成します。

新規ディレクトリーの作成

それぞれのコネクター・インスタンスごとにコネクター・ディレクトリーを作成する必要があります。このコネクター・ディレクトリーには、次の名前を付けなければなりません。

```
ProductDir¥connectors¥connectorInstance
```

ここで connectorInstance は、コネクター・インスタンスを一意的に示します。

コネクターに、コネクター固有のメタオブジェクトがある場合、コネクター・インスタンス用のメタオブジェクトを作成する必要があります。メタオブジェクトをファイルとして保管する場合は、次のディレクトリーを作成して、ファイルをそこに格納します。

```
ProductDir¥repository¥connectorInstance
```

ビジネス・オブジェクト定義の作成

各コネクター・インスタンスのビジネス・オブジェクト定義がプロジェクト内にまだ存在しない場合は、それらを作成する必要があります。

1. 初期コネクターに関連付けられているビジネス・オブジェクト定義を変更する必要がある場合は、適切なファイルをコピーし、Business Object Designer Express を使用してそれらのファイルをインポートします。初期コネクターの任意のファイルをコピーできます。変更を加えた場合は、名前を変更してください。
2. 初期コネクターのファイルは、次のディレクトリーに入っていなければなりません。

   ```
   ProductDir¥repository¥initialConnectorInstance
   ```

   作成した追加ファイルは、`ProductDir¥repository` の適切な `connectorInstance` サブディレクトリー内に存在している必要があります。

コネクター定義の作成

Connector Configurator Express 内で、コネクター・インスタンスの構成ファイル (コネクター定義) を作成します。これを行うには、以下のステップを実行します。

1. 初期コネクターの構成ファイル (コネクター定義) をコピーし、名前変更します。
2. 各コネクター・インスタンスが、サポートされるビジネス・オブジェクト (および関連メタオブジェクト) を正しくリストしていることを確認します。
3. 必要に応じて、コネクター・プロパティーをカスタマイズします。

始動スクリプトの作成

始動スクリプトは以下のように作成します。

1. 初期コネクターの始動スクリプトをコピーし、コネクター・ディレクトリーの名前を含む名前を付けます。

   dirname
2. この始動スクリプトを、27 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. 始動スクリプトのショートカットを作成します。
4. 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。

これで、ご使用の統合サーバー上でコネクターの両方のインスタンスを同時に実行することができます。

コネクターの始動

コネクターは、**コネクター始動スクリプト**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ーのランタイム・ディレクトリーに存在していなければなりません。

ProductDir¥connectors¥*connName*

ここで、*connName* はコネクターを示します。始動スクリプトの名前は、表 5 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 5. コネクターの始動スクリプト

オペレーティング・システム	始動スクリプト
Windows	start_*connName*.bat

コネクター始動スクリプトは、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 Express」>「アダプター」>「コネクター」>「*ご使用のコネクター名 (your_connector_name)*」を選択します。

  デフォルトでは、プログラム名は「IBM WebSphere Business Integration Express」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、ご使用のコネクターへのデスクトップ・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* **ICS** [-c*configFile*]

    ここで、*connName* はコネクターの名前です。
- System Monitor から。

  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、コネクターが始動します。

コマンド行の始動オプションなどのコネクターの始動方法の詳細については、「*システム管理ガイド*」を参照してください。

コネクターの停止

コネクターを停止する方法は、以下に示すように、コネクター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ーを始動した場合は、コネクター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、そのコネクター用の別個の「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、コネクターが停止します。
- System Monitor を使用します。

  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。
- 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。

第 3 章 コネクター用のビジネス・オブジェクトについて

Adapter for PeopleSoft のコネクター・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、コネクター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ーは、サポートする各ビジネス・オブジェクトを、コネクター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、属性プロパティーの設定、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。コネクターはメタデータ主導型のため、コネクター・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。コネクターはビジネス・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付けて操作します。

コネクターには、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、コネクター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、コネクター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、コネクター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、コネクターによるビジネス・オブジェクトの処理方法と、コネクター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。

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

分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。

コネクターは、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、コネクター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。コネクターは 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場合、次の状態が考えられます。

- 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。

- 個々のビジネス・オブジェクトが、対応するコレクションが持つフィールドよりも多くの単純属性を持つ可能性があります (つまり、ビジネス・オブジェクトの一部の属性は、コレクション内で表されません)。コレクション内に表現されない属性は、アプリケーション固有情報がないか、またはデフォルト値が設定されます。

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

コネクターには、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、コネクター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、コネクターの作成はすべて 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、コネクターがサポート・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、38 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』
- 37 ページの『ビジネス・オブジェクトの例』
- 37 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 6 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。

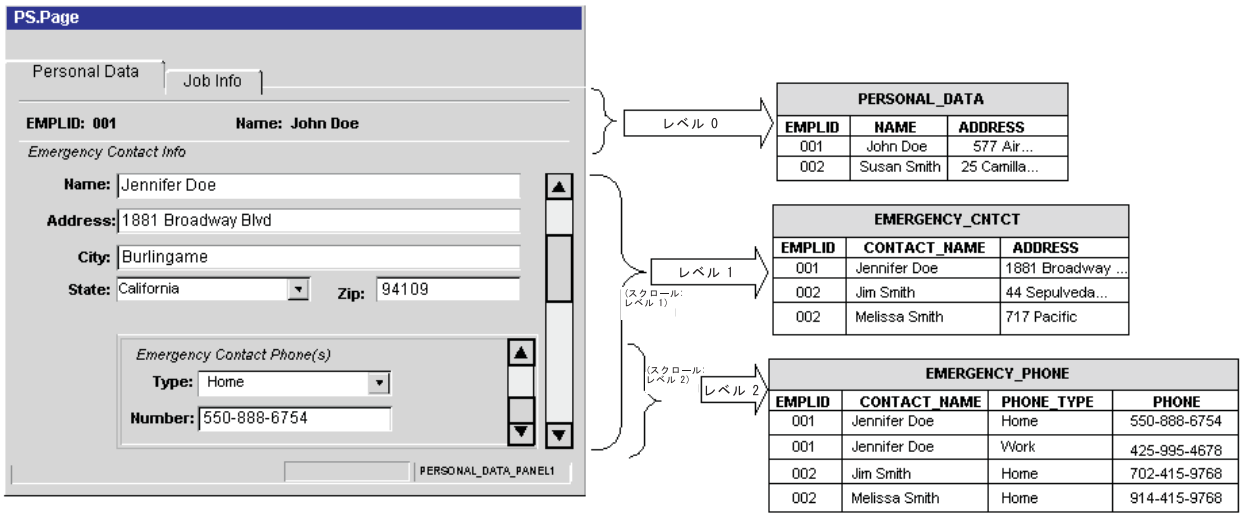

図 6. コンポーネントの例

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ので、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ので、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』
- 35 ページの『隠しフィールド』
- 35 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。コネクター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・プロパティー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 のデータ履歴を戻すと仮定します。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 を持つ一連の行を戻す場合、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・プロセッサー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、コネクター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、コネクター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。コネクター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、54 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 7 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。

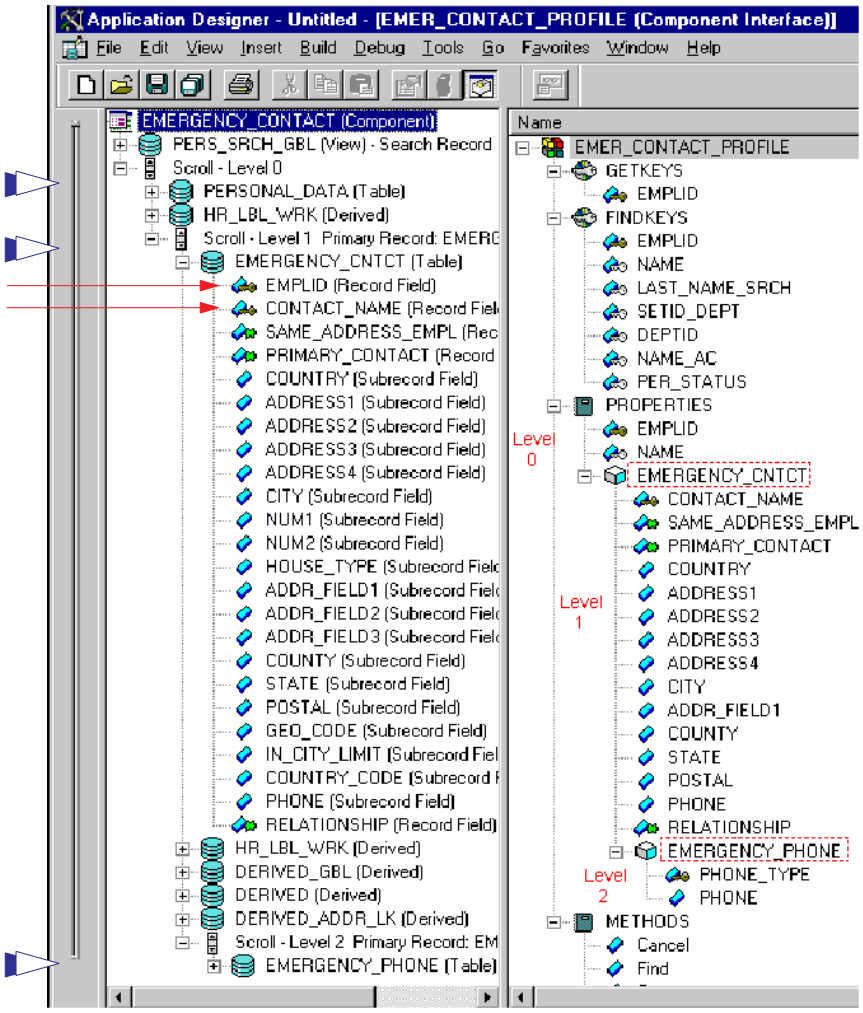

図 7. コンポーネント・インターフェースの例

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、コネクターはコンポーネント・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にのみ関連するすべての PeopleCode イベントにも適用されます。

したがって、コネクターのコンポーネント・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、コネクター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ー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 8 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。

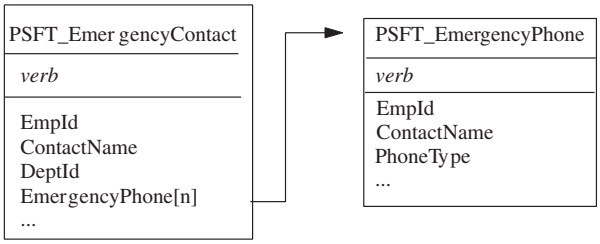

図 8. ビジネス・オブジェクトの例

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。初めの 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 Express 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、50 ページの『ビジネス・オブジェクトの属性プロパティー』と 52 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、 57 ページの『第 4 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ーのサポート・ビジネス・オブジェクトのリストに追加する。使用する InterChange Server Express 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』
- 『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。

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ーに存在することを確認する。

%ProductDirS% directory:

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

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、19 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 プロパティーに格納されます。詳細については、53 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』
- 『setFieldName() メソッド』
- 40 ページの『getCollectionName() メソッド』
- 40 ページの『CurrentItem() メソッド』
- 40 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 get*FieldName*() 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 7 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、コネクターは getEmpId()、getName()、および getDeptId() メソッドを使用します。値を取得後、コネクターはそれらの値を 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、コネクター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_CNTCT 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、コネクターは最初に 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、40 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 set*FieldName*() メソッドがあり、このメソッドによりコネクターが、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、コネクター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、コネクターは EmerContactProfile クラス内の getEmergencyCntct() メソッドを使用します。コネクター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、コネクターは 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、コネクターは 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、コネクター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、53 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、コネクターは 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、コネクター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、コネクターは EmerContactProfile クラス内の CurrentItem() メソッドを使用します。コネクターは、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、コネクターは EmerContactProfile クラス内の Item(*index*) メソッドを使用します。コネクターは、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。

- 41 ページの『変更後イメージとデルタ』で、用語を定義し、コネクターの変更後イメージの使用方法を説明します。
- 42 ページの『ビジネス・オブジェクト要求の動詞の処理』で、コネクターがビジネス・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。
- 50 ページの『データのコミット』で、コネクター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。このコネクターは、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。コネクター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、コネクターが 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、コネクター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。

コネクターが子ビジネス・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 9 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。

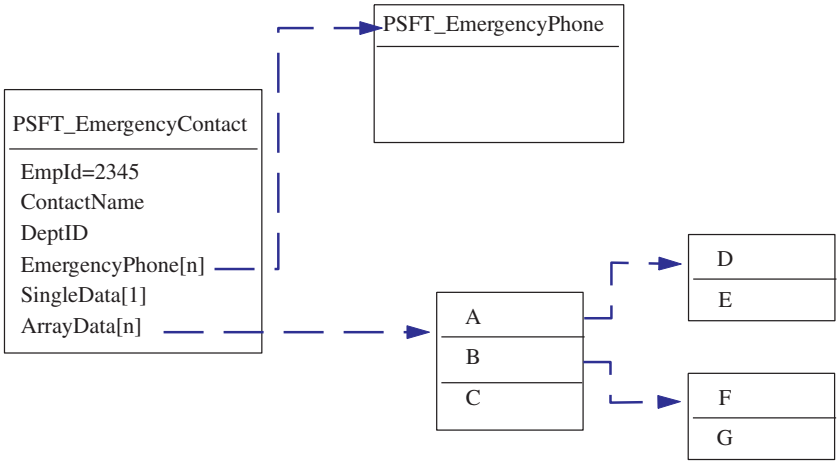

図 9. 更新前のデータの状態

図 10 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。

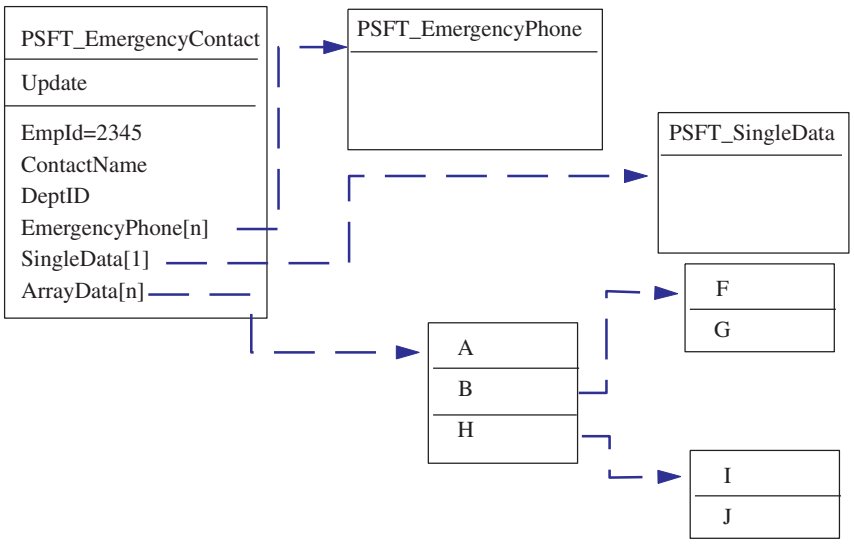

図 10. 更新要求で表わされるデータ

更新を処理するために、コネクターはコンポーネント・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。

コネクターは、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ー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ーが、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ー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、54 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、コネクター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。コネクターは階層ビジ

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、コネクターは 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、コネクターは特定のビジネス・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、コネクターはビジネス・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、コネクター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 のいずれかの状況を戻します。

コネクターは、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 Set*FieldName*(*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ーにビジネス・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、コネクター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 プロパティーには、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 プロパティー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 プロパティーを使用して、ストリング NEXT を指定する場合は、コネクター

の UseDefaults プロパティーの評価が true である必要があります。詳細については、27 ページの『UseDefaults』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 プロパティー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、50 ページの『Required プロパティー』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、コネクターは子ビジネス・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 プロパティーの評価が 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、コネクターはエラーを戻し、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、コネクターは属性を無視します。

3. コネクターは、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。
   - コネクターが固有 ID を生成する場合、このメソッドの実行時に生成されます。
   - アプリケーションが固有 ID を生成する場合、コネクター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 Express が統合ブローカーの場合、InterChange Server Express 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、コネクターが FAIL 戻りコードを送信します。

属性プロパティーの詳細については、50 ページの『ビジネス・オブジェクトの属性プロパティー』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、54 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 のいずれかの状況を戻します。

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. InterChange Server Express 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。

   コネクターはソース・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、コネクター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、コネクターは 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、コネクターは 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ー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、コネクター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、コネクターは検索時に、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 InterChange Server Express 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 という状況を戻します。

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. InterChange Server Express 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。

   コネクターは、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、 54 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、コネクター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、コネクターは 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、コネクターは 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ー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、コネクター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、コネクターは検索時に、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 InterChange Server Express 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。

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 のいずれかの状況を戻します。

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更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、コネクターはコンポーネント・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、コネクター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 プロパティーには、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 プロパティー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 プロパティーを使用して、ストリング NEXT を指定する場合は、コネクターの UseDefaults プロパティーの評価が true である必要があります。詳細については、27 ページの『UseDefaults』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 プロパティー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。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、コネクターは子ビジネス・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、コネクターは 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。コネクターは、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、54 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す。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。
   - 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、コネクターはコンポーネント・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、47 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 プロパティー

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、このプロパティーに `true` を設定します。

注: コネクターは、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。

コネクター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、43 ページの『作成操作』および 47 ページの『更新操作』を参照してください。

Required プロパティー

`Required` プロパティーで、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ーが指定されていると、コネクター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。

コネクター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、コネクターは、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。

コネクター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、コネクターは検索操作を失敗させます。

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

Max length プロパティー

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

AppSpecificInfo

このプロパティーの詳細については、54 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 プロパティー

このプロパティーで、属性に値が設定されていない場合にコネクター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。コネクターは、子ビジネス・オブジェクトを表す属性に対して、このプロパティーを評価しません。作成操作では、コネクターは、その UseDefaults プロパティーの評価が true の場合にのみ、このプロパティーの値を使用します。詳細については、27 ページの『UseDefaults』を参照してください。

コネクターは、次が true の場合、このプロパティー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 プロパティーが指定されている。
- Default Value プロパティーの値が、ストリング NEXT である。
- コネクターの UseDefaults プロパティーの評価が 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、43 ページの『作成操作』および 47 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えられます。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、コネクターが予期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 6 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使用できる機能の概要を説明します。

表 6. 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	機能性
ビジネス・オブジェクト全体	次を指定します。 • 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。 • アプリケーションで固有 ID 値を生成する、またはコネクターで固有 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	コネクターは、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]
```

表 7 で、これらのパラメーターについて説明します。

表 7. 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。
EFFDT (isEffectiveDated)	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。 • EFFDT パラメーターの評価が true の場合、コネクターは 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、コネクターは 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、コネクターは将来の日付がリストされていると、それを戻します。 デフォルト値は true です。
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。このプロパティーは、通常、パフォーマンスを向上するために使用されます。 • true に設定されていると、コネクターは、プロパティーまたは属性の値が変更されるたびに処理します。つまり、コネクターが各属性について set*PropertyName*() を呼び出した後で、即時にすべてのオンライン Field PeopleCode を起動します。 • false 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、コネクターがコンポーネント・インターフェースで Save() 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。 デフォルト値は true です。

表 7. 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
GetHistoryItems	コネクターが検索するデータ量を決定します。 • true に設定されていると、コネクターは、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。 • false に設定されていると、コネクターは現在のデータ (有効な行) のみを検索します。 デフォルト値は true です。
GetDummyRows	PeopleTools 8.4 以上のバージョンの GetDummyRows プロパティーをサポートします。このプロパティーは、ご使用の PeopleTools がこれよりも前のバージョンである場合は設定しないでください。このプロパティーの詳細については、PeopleSoft の資料を参照してください。
InsAtOldestEffDtPos	このパラメーターは、getEffectiveItemNum() 呼び出しによって有効期限行の挿入位置に -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、getEffectiveItemNum() 呼び出しによって有効期限行の挿入位置に -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。
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 プロパティーの詳細については、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=get<i>FieldName</i>:set=set<i>FieldName</i>: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
```

子を保存する必要の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 8 で、各名前値パラメーターを説明します。

表 8. 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=get*FieldName*	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=set*FieldName*	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=[truelfalse]	このパラメーターが false 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、コネクターはエラー・メッセージを送信します。 このパラメーターが true 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、コネクターの UseDefaults プロパティーの評価が true である必要があります。詳細については、27 ページの『UseDefaults』を参照してください。 キー値を varchar 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 UID=True に設定してください。キー値が varchar 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。 アプリケーションで固有 ID を生成する場合、コネクターは、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。 統合ブローカーが InterChange Server Express である場合、コネクターは検索した ID を使用してビジネス・オブジェクトを相互参照します。 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 false 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。

表 8. 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]	このパラメーターが true に設定されて、属性の Key プロパティーと Required プロパティーの評価が true の場合、コネクターは検索または更新用の検索操作時に、キーの一部としてこの属性を組み込みます。 コネクター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、コネクターはキーとして、その Key プロパティーおよび Required プロパティーの評価が true 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、コネクターは検索操作を実行する際に、その GetKey パラメーターの評価が true のすべての属性をキーに追加します。
KeepRelationship=[true|false]	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ーが削除するかどうかを指定します。 • 削除を防止するには、true に設定します。 • 削除を許可するには、false に設定します。 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、コネクターは単一の子ビジネス・オブジェクトを含む PSFT_EmergencyContact 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。EmergencyPhone[n] 属性の KeepRelationship の評価が true の場合、コネクターは、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。 ただし、KeepRelationship の評価が false の場合、コネクターはソース・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。
findKey=[true|false]	このパラメーターが true に設定されている場合、コネクター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。
bigDec=[true|false]	このパラメーターによって、BigDecimal 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 BigDecimal 属性のこのパラメーターには true を設定してください。
EFFDT=[true|false]	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。コネクターは、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 EFFDT が true 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。
EFFDTSEQ=[true|false]	このパラメーターによって、getEffectiveItemNum 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。コネクターは、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。

第 4 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、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。コネクターは 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

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

PeopleSoftODA をインストールするには、IBM WebSphere Business Integration Server Express Plus Adapters 用のインストーラーを使用します。「*WebSphere Business Integration Server Express インストール・ガイド*」を参照してください。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_*ll_TT*.txt ファイル (言語 (*_ll*) 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)

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。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。

- 13 ページの『コネクターを使用するために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
- 17 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、38 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 9 で説明されている値が正しいことを確認します。

表 9. 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	**WINDOWS:** set AGENT = %*ProductDir*S%¥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、「*WebSphere Business Integration Server Express インストール・ガイド*」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 Express を起動します。
3. Business Object Designer Express 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

このスクリプトを使用して PeopleSoftODA を起動します。

•

Windows

```
start_PeopleSoftODA.bat
```

Windows の終り

PeopleSoftODA の構成と実行には、Business Object Designer Express を使用します。Business Object Designer Express 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。このコネクター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。

62 ページの図 11 に、ODA を選択して実行する Business Object Designer Express 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ー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ー下の ¥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 Express 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 Express 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 Express 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 Express 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、62 ページの『初期化プロパティーの構成』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 10 で、これらの値を説明します。

表 10. 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 のプロパティー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	• そのすべてのプロパティーに関する 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、62 ページの『初期化プロパティーの構成』を参照してください。

Business Object Designer Express での PeopleSoftODA の使用

このセクションでは、Business Object Designer Express 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 Express 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 Express を起動させ、ODA を構成して実行します。Business Object Designer Express 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 Express 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 Express 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 Express 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。62 ページの図 11 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. 『ODA の選択』
2. 62 ページの『初期化プロパティーの構成』
3. 64 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. 65 ページの『オブジェクトの選択の確認』
5. 66 ページの『定義の生成』。必要であれば 67 ページの『追加情報の入力』
6. 68 ページの『定義の保管』

ODA の選択

図 11 に、Business Object Designer Express 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。

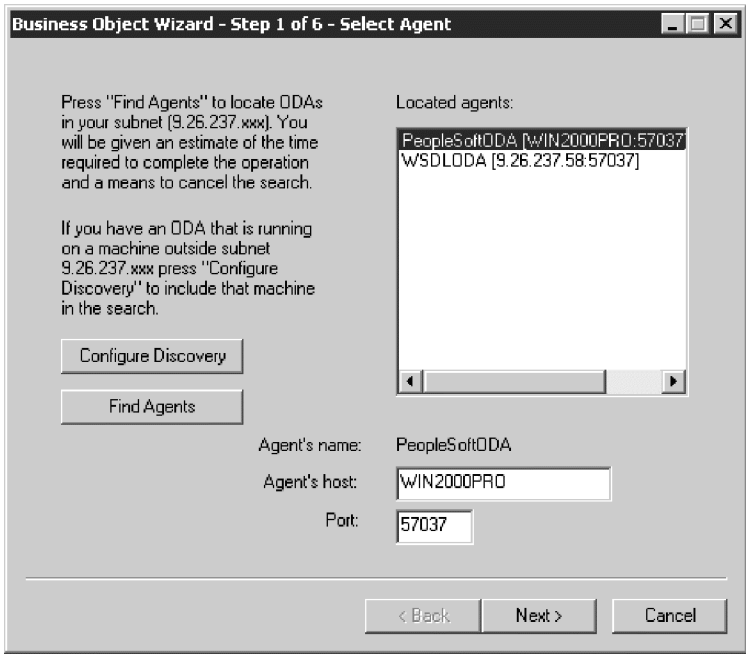

図 11. ODA の選択

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 Express 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 Express 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。

初期化プロパティーの構成

Business Object Designer Express は、PeopleSoftODA と初めて通信する際に、一連の初期化プロパティーの入力を要求します。その画面を図 12 に示します。これらのプロパティーは、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。

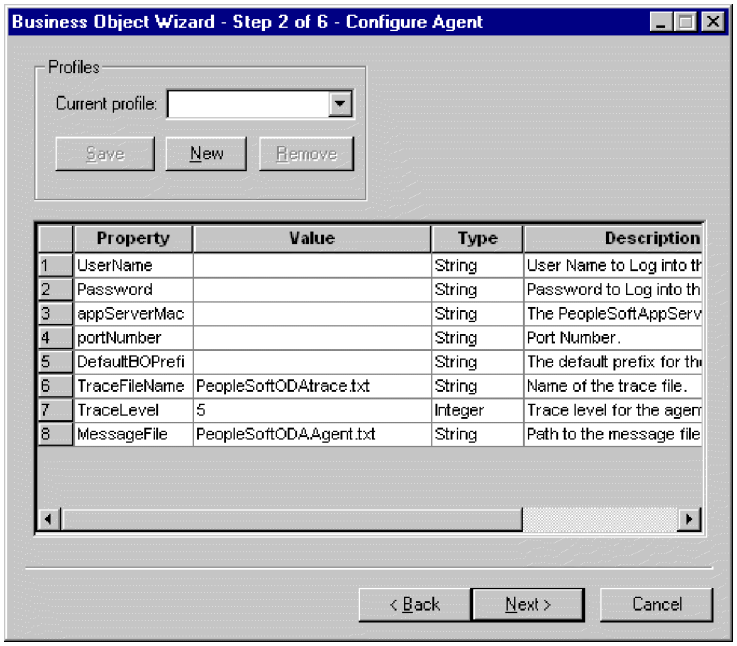

図 12. エージェント初期化プロパティーの構成

PeopleSoftODA プロパティーの構成を表 11 に示します。

表 11. PeopleSoftODA プロパティー

行番号	プロパティー名	プロパティー・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ー (_) で始まることはできません。必要であれば、Business Object Designer Express によりビジネス・オブジェクト・プロパティー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、67 ページの『追加情報の入力』を参照してください。

表 11. PeopleSoftODA プロパティー (続き)

行番号	プロパティー名	プロパティー・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、PeopleSoftODAtrace.txt という名前のトレース・ファイルを生成します。このプロパティーを使用して、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、59 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・プロパティーの値は 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。このプロパティーを使用して、既存のファイルの確認や指定をします。

重要: Business Object Designer Express 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 Express 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 Express 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 Express は以前のステップで構成されたプロパティーを使用して、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 Express 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 13 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。

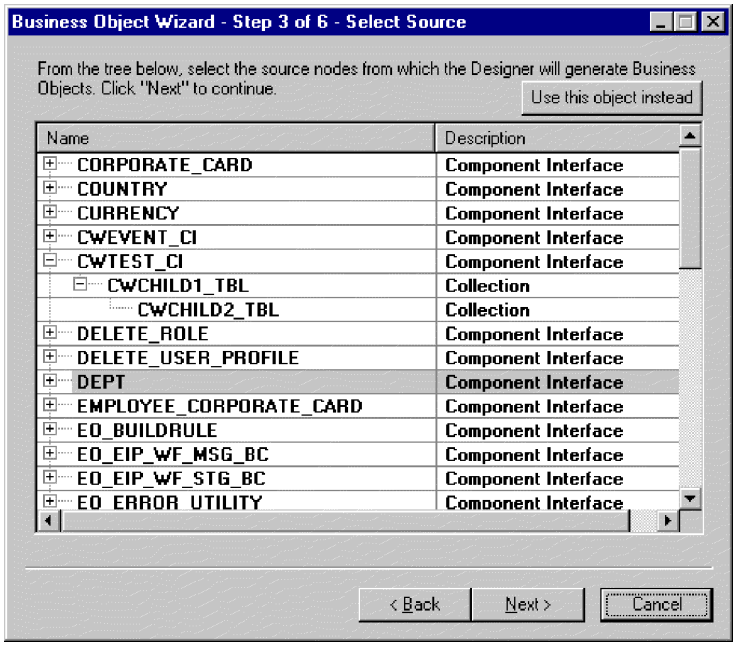

図 13. ノードが展開されたコンポーネント・インターフェースのツリー

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 Express 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 14 にこのダイアログ・ボックスを示します。

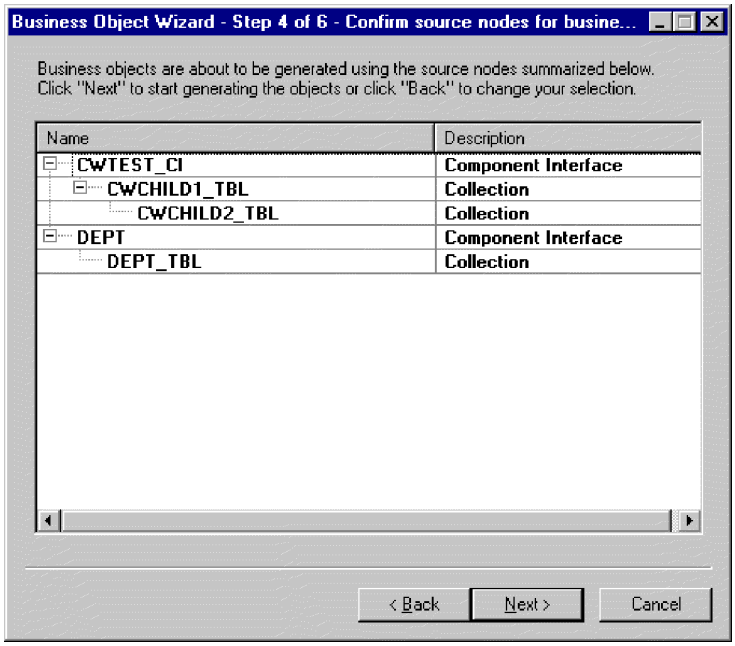

図 14. オブジェクトの選択の確認

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 Express 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 15 にこのダイアログ・ボックスを示します。

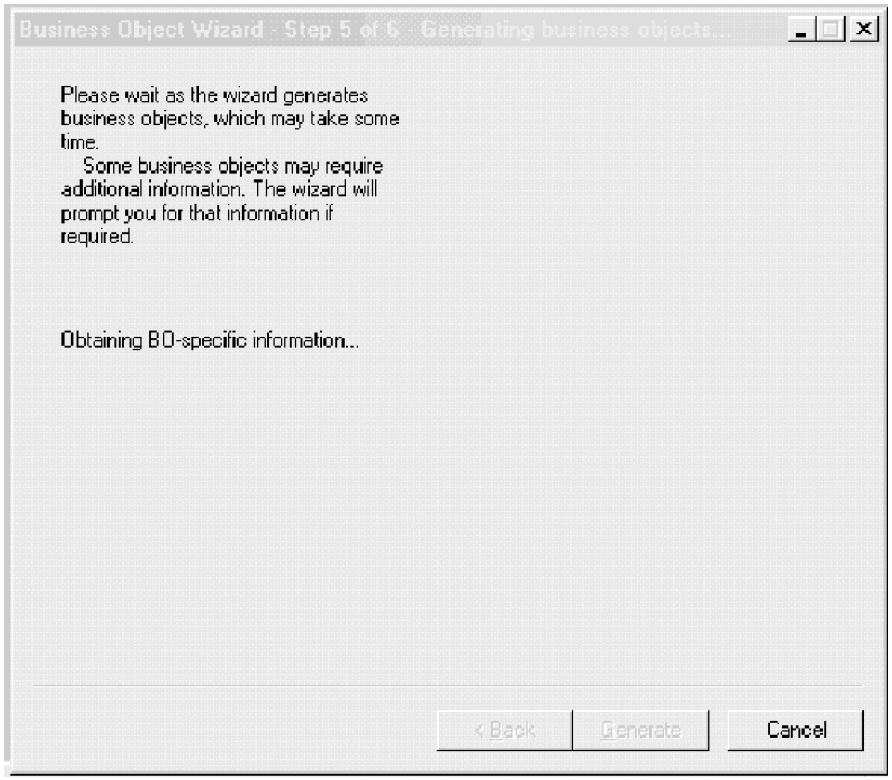

図 15. 定義の生成

追加情報の入力

PeopleSoftODA に追加情報が必要な場合、Business Object Designer Express により、情報の入力を要求する「BO プロパティー」ウィンドウが表示されます。図 16 にこのダイアログ・ボックスを示します。

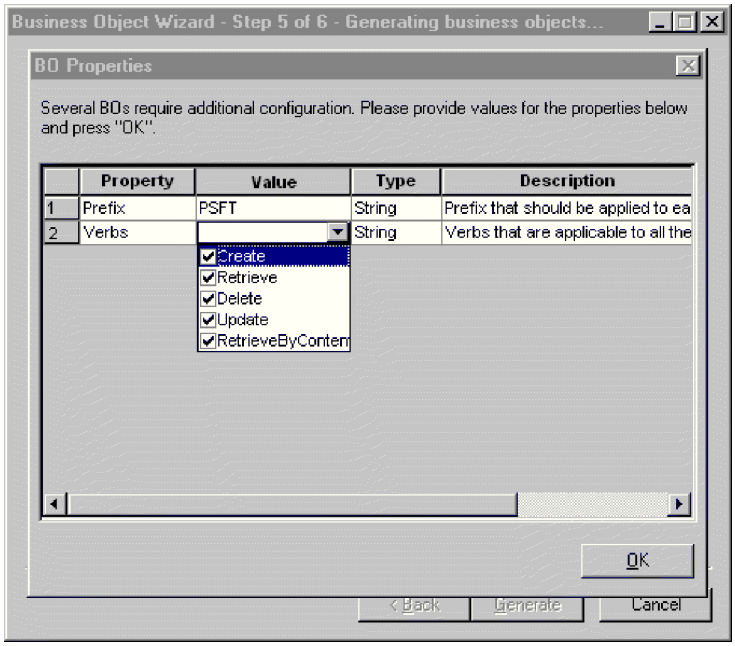

図 16. 追加情報の入力

「BO プロパティー」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 (63 ページの図 12) で *DefaultBOPrefix* プロパティー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 プロパティー」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 プロパティー」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 Express 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、定義をサーバーまたはファイルに保管するか、あるいは Business Object Designer Express 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 17 にこのダイアログ・ボックスを示します。

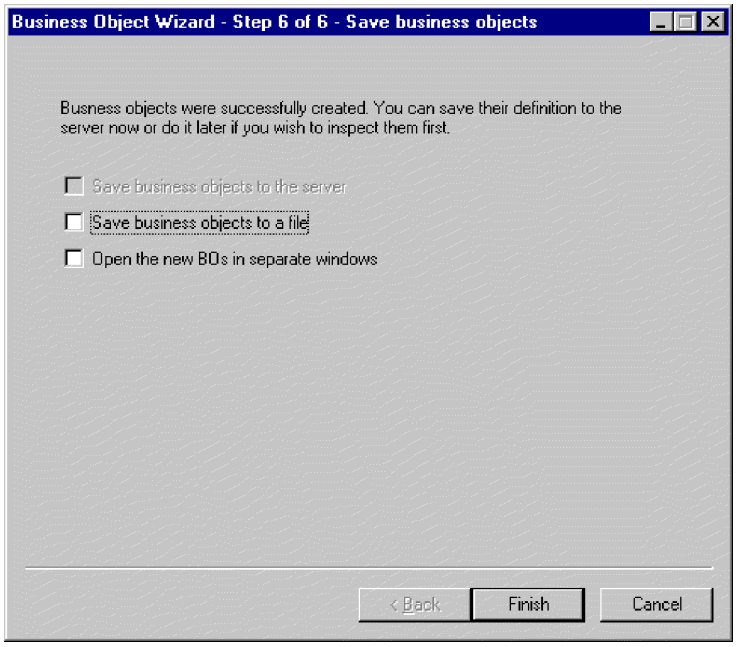

図 17. ビジネス・オブジェクト定義の保管

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ーの単純属性。
- 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 プロパティー」ウィンドウ (68 ページの図 16 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ー』
- 70 ページの『属性プロパティー』

ビジネス・オブジェクト・レベルのプロパティー

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 12 で、これらのパラメーターについて説明します。

表 12. 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 プロパティー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。他のプロパティー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、残りのプロパティーの値を指定する必要があります。これらのプロパティーの詳細については、53 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。

属性プロパティー

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ーについて説明します。属性の詳細については、50 ページの『ビジネス・オブジェクトの属性プロパティー』を参照してください。

Name プロパティー

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ーから属性の名前を派生させます。

Data Type プロパティー

属性のタイプを設定する場合、PeopleSoftODA は、表 13 に示すように、プロパティー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 13. 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 13. データ型の対応 (続き)

Float	Float	
Number	Integer	

注: プロパティーのデータ型が表 13 に示す中のいずれでもない場合、PeopleSoftODA はプロパティーをスキップして、プロパティーを処理できないことを示すメッセージを表示します。

Cardinality プロパティー

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 プロパティー

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 プロパティー

プロパティーがコンポーネント・インターフェース内の CreateKey の場合、PeopleSoftODA はこのプロパティーを true に設定します。プロパティーがコンポーネント・インターフェース内の GetKey の場合、PeopleSoftODA はこのプロパティーに 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 プロパティー

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

IsRequired プロパティー

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

AppSpecificInfo プロパティー

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=get<i>FieldName</i>:set=set<i>FieldName</i>:UID=[true|false]:GetKey=[true|false]:KeepRelationship=[true|false]
```

PeopleSoftODA は、表 14 で説明するように、属性に関連するプロパティーのみを生成します。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 14. 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ー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。コネクター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=get`*`PropertyName`*	`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ー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ー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。コネクターはこの方法で、更新された属性の値を検索します。
`get=get`*`CollectionName`*	`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。コネクターはこの方法で、コレクションの値を検索します。
`set=set`*`PropertyName`*	`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ー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ー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。コネクターはこの方法で、属性の値を更新します。

注: Business Object Designer Express 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、54 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 14 で説明されているパラメーターの使用例については、72 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』を参照してください。

動詞

注: PeopleSoftODA は、「BO プロパティー」ウィンドウ (68 ページの図 16 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 = 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=getEmplid:set=setEmplid:keepRelationship=false:uid=true:
findKey=true:getKey=true:createKey=true     IsRequiredServerBound = false
    [End]     [Attribute]     Name = EMPL_RCD     Type = String
Cardinality = 1     MaxLength = 1     IsKey = true     IsForeignKey = false
    IsRequired = true     AppSpecificInfo =
get=getEmplRcd:set=setEmplRcd:keepRelationship=false:uid=true:
findKey=true:getKey=true:createKey=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=getEmplid0:set=setEmplid0: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

次に示す例は 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 &CWPRIORITY1; Component string
&CONNID; &BONAME1 = "Psft_Dept"; &KEYLIST1 =
"DEPT_TBL.SETID:DEPT_TBL.DEPTID"; &CWPRIORITY1 = 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, &CWPRIORITY1, &CONNID);    End-If;
```

ビジネス・オブジェクト定義内の情報の変更

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、31 ページの『第 3 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 Express 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ーに再ロードするには、Business Object Designer Express を使用します。

第 5 章 トラブルシューティングとエラー処理

この章では、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

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` ユーティリティーを使用して、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` ユーティリティーを使用して、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、38 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、15 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ー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。

処理時の問題

コネクター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ー値の検索。

このテストで、PeopleSoft API とコネクター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、コネクター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

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 15 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 15. 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	• 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	• 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 `gotApplEvent()` かまたは `consumeSync()` から InterChange Server Express 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。

表 15. 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 3	• 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。
レベル 4	• 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。 • コネクターのプロセス・フローをトレースできる関数を、コネクター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。コネクター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す。
レベル 5	• コネクターの初期設定を示すメッセージ。例えば、InterChange Server Express から検索した各構成プロパティー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、コネクターのログ・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。 • コネクターがビジネス・オブジェクトを処理する前 (コネクターが受け取ったままの状態を表示) と処理を完了した後 (コネクターが InterChange Server Express 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。 • コネクター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

コネクター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

コネクター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、コネクターは InterChange Server Express 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ー

この付録では、WebSphere InterChange Server Express で動作する、WebSphere Business Integration Server Express のアダプターに含まれるコネクター・コンポーネントの標準構成プロパティーについて説明します。

コネクターによっては、一部の標準プロパティーが使用されないことがあります。Connector Configurator Express から統合ブローカーを選択すると、ご使用のアダプターに対して構成する必要がある標準プロパティーのリストが表示されます。

コネクター固有のプロパティーの詳細については、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。

標準コネクター・プロパティーの構成

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ーがあります。

- 標準構成プロパティー
- コネクター固有の構成プロパティー

このセクションでは、標準構成プロパティーについて説明します。コネクター固有の構成プロパティーについては、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 Express の使用

コネクター・プロパティーの構成は Connector Configurator Express から行います。Connector Configurator Express には、System Manager からアクセスします。Connector Configurator Express の使用方法の詳細については、付録 B『Connector Configurator Express』を参照してください。

プロパティー値の設定と更新

プロパティー・フィールドのデフォルトの長さは 255 文字です。

コネクターは、以下の順序に従ってプロパティーの値を決定します (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. リポジトリー
3. ローカル構成ファイル
4. コマンド行

コネクターは、始動時に構成値を取得します。実行時セッション中に 1 つ以上のコネクター・プロパティーの値を変更する場合は、プロパティーの**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。標準コネクター・プロパティーには、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。
- **コンポーネント再始動**
  System Manager でコネクター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。
- **エージェント再始動**
  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。

特定のプロパティーの更新方法を確認するには、「Connector Configurator Express」ウィンドウ内の**「更新メソッド」**列を参照するか、次に示すプロパティーの要約の表の「更新メソッド」列を参照してください。

標準プロパティーの要約

表 16 は、標準コネクター構成プロパティーの早見表です。標準プロパティーの依存関係は RepositoryDirectory に基づいているため、コネクターによっては使用されないプロパティーがあり、使用する統合ブローカーによってプロパティーの設定が異なる可能性があります。

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

表 16. 標準構成プロパティーの要約

プロパティー名	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 IDL
AgentTraceLevel	0 から 5	0	動的	
ApplicationName	アプリケーション名	コネクター・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	ICS		
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	

表 16. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント再始動	Repository Directory は <REMOTE>
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
ControllerStoreAndForwardMode	true または false	true	動的	Repository Directory は <REMOTE>
ControllerTraceLevel	0 から 5	0	動的	Repository Directory は <REMOTE>
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	コンポーネント再始動	JMS トランスポートのみ
DeliveryTransport	IDL または JMS	IDL	コンポーネント再始動	
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> でなければならない
EnableOidForFlowMonitoring	true または false	false	コンポーネント再始動	
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	crossworlds.queue.manager	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>
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>
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>

表 16. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
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	
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE>
MessageFileName	パスまたはファイル名	InterchangeSystem.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 でなければならない。
OADAutoRestartAgent	true または false	false	動的	Repository Directory は <REMOTE>
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE>
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE>
PollEndTime	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	
PollFrequency	正整数 (単位: ミリ秒) no (ポーリングを使用不可にする) key (コネクターのコマンド・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・リポジトリーの場所		エージェント再始動	<REMOTE> に設定する

表 16. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
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 は JMS
RestartRetryCount	0 から 99	3	動的	
RestartRetryInterval	適切な正数 (単位: 分): 1 から 2147483547	1	動的	
SourceQueue	有効な JMS キュー名	*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	有効な JMS キュー名	*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	有効な JMS キュー名	*CONNECTORNAME*/SYNCHRONOUSRESPONSEQUEUE	コンポーネント再始動	Delivery Transport は JMS
WireFormat	CwBO	CwBO	エージェント再始動	

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

統合ブローカーからコネクタ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/ADMININQUEUE です。

AdminOutQueue

コネクター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

AgentConnections プロパティーは、orb.init[] により開かれる ORB 接続の数を制御します。

デフォルトでは、このプロパティーの値は 1 に設定されます。このデフォルト値を変更する必要はありません。

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。コネクターは、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

コネクター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。コネクターを実行する前に、このプロパティー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を指定します。ICS を指定する必要があります。

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 ベースのコネクターでは、このプロパティーは使用しません。C++ ベースのコネクターでは、現在、このプロパティーに 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・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・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 Express の使用方法に関する付録を参照してください。

ConcurrentEventTriggeredFlows

コネクター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。

このプロパティーを 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ーが、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 プロパティーの値を増加して、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、またはコネクター・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 構成プロパティーに、1 より大きい値を設定します。

ConcurrentEventTriggeredFlows プロパティーは、順次に実行される単一スレッド処理であるコネクターのポーリングでは無効です。

ContainerManagedEvents

このプロパティーにより、JMS イベント・ストアを使用する JMS 対応コネクターが、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。

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。

デフォルト値は No value です。

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ーも構成する必要があります。

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

また、MimeType、DHClass、および DataHandlerConfigMOName (オプショナル) プロパティーを設定したデータ・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ーの値を設定するには、Connector Configurator Express の**「データ・ハンドラー」**タブを使用します。「データ・ハンドラー」タブ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。

ControllerStoreAndForwardMode

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

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 WebSphere InterChange Server Express 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、IDL (CORBA IIOP) または JMS (Java Messaging Service) です。デフォルトは IDL です。

DeliveryTransport プロパティーに指定されている値が IDL である場合、コネクターは、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。

JMS

Java Messaging Service (JMS) を使用しての、コネクターとクライアント・コネクター・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すると、jms.MessageBrokerName、jms.FactoryClassName、jms.Password、jms.UserName などの追加の JMS プロパティーが Connector Configurator Express 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ーです。

重要: WebSphere InterChange Server Express で動作しているコネクターで JMS トランスポート機構を使用すると、メモリー制限が発生することがあります。

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。

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。

EnableOidForFlowMonitoring

このプロパティーを true に設定すると、アダプター・フレームワークは、フロー・モニターを使用できるようにするため、着信 **ObjectEventId** を外部キーとしてマークします。

デフォルト値は false です。

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)。

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)。

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)。

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、このコネクター・プロパティーを必ず 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として選択するときは (DeliveryTransport を参照)、このコネクター・プロパティーを必ず 設定してください。

デフォルト値は crossworlds.queue.manager です。

jms.NumConcurrentRequests

コネクター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。このプロパティー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。このプロパティーの値はオプションです。

デフォルトはありません。

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。このプロパティーの値は、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。

ll	2 文字の言語コード (普通は小文字)
TT	2 文字の国または地域コード (普通は大文字)
codeset	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・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・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 Express の使用方法に関する付録を参照してください。

デフォルト値は en_US です。コネクターがグローバル化に対応していない場合、このプロパティーの有効な値は en_US のみです。特定のコネクター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ーのバージョン・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/wicserver/infocenter

LogAtInterchangeEnd

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ー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・バッファー内のイベントの最大数。このプロパティーは、フロー制御で使用されます。

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

コネクター・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 ¥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。

コネクター・メッセージ・ファイルが存在しない場合は、コネクターは 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: 特定のコネクターについて、コネクター独自のメッセージ・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ーズ・ガイドを参照してください。

MonitorQueue

コネクター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。このプロパティーは、DeliveryTransport プロパティー値が 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

コネクター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ーを再始動したり、System Monitor からリモート・コネクター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、このプロパティーを 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)*」を参照してください。

デフォルト値は false です。

OADMaxNumRetry

異常シャットダウンの後で MQ により起動される OAD がコネクター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。このプロパティーを有効にするためには、OADAutoRestartAgent プロパティーを true に設定する必要があります。

デフォルト値は 1000 です。

OADRetryTimeInterval

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。コネクター・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、コネクター・コントローラーはコネクター・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 プロパティーで指定された回数だけ繰り返します。このプロパティーを有効にするためには、OADAutoRestartAgent プロパティーを 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。

このプロパティー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

ポーリング・アクション間の時間の長さです。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。
- ワード key。コネクターは、コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力されたときに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。コネクター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: 一部のコネクターでは、このプロパティーの使用が制限されています。このプロパティーが使用されるかどうかを特定のコネクターについて判別するには、該当する Adapter ガイドのインストールと構成についての章を参照してください。

PollQuantity

コネクター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ー固有のポーリング数設定プロパティーがある場合、標準プロパティーの値は、このコネクター固有のプロパティーの設定値によりオーバーライドされます。

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。

このプロパティー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

WebSphere InterChange Server Express からコネクターへビジネス・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 CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

コネクターが XML スキーマ文書を読み取るリポジトリーの場所です。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。

この値は <REMOTE> に設定する必要があります。これは、コネクターが InterChange Server Express リポジトリーからこの情報を取得するためです。

ResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

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ー・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。WebSphere InterChange Server Express は、要求を送信した後、JMS 応答キューで応答メッセージを待機します。

RestartRetryCount

コネクターによるコネクター自体の再始動の試行回数を指定します。このプロパティーを並列コネクターに対して使用する場合、コネクター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

コネクターによるコネクター自体の再始動の試行間隔を分単位で指定します。このプロパティーを並列コネクターに対して使用する場合、コネクター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。

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 対応コネクター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ー・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、 89 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、コネクター・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、コネクター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、コネクター・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。コネクター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ー・フレームワークにデリバリーします。このキューは、コネクター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。

コネクター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。コネクターは、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。設定値は CwBO です。

付録 B. Connector Configurator Express

この付録では、Connector Configurator Express を使用してアダプターの構成プロパティー値を設定する方法について説明します。

この付録では、次のトピックについて説明します。

Connector Configurator Express の概要

Connector Configurator Express では、WebSphere InterChange Server Express で使用するアダプターのコネクター・コンポーネントを構成できます。

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

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成する。
- **コネクター構成ファイル**を作成します。インストールするコネクター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ーを設定する。
  場合によっては、コネクター・テンプレートでプロパティー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、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、ロギングとトレース、およびデータ・ハンドラーに関するパラメーターを指定する必要があります。

コネクター構成プロパティーには、標準の構成プロパティー (すべてのコネクターがもつプロパティー) と、コネクター固有のプロパティー (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ーで必要なプロパティー) とが含まれます。

標準プロパティーは、すべてのコネクターで使用されるので、新規に定義する必要はありません。構成ファイルを作成すると、Connector Configurator Express によって標準プロパティーがそのファイルに挿入されます。ただし、Connector Configurator Express で各標準プロパティーの値を設定する必要があります。

標準プロパティーの範囲は、ブローカーと構成によって異なる可能性があります。特定のプロパティー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ーがあります。Connector Configurator Express の「標準のプロパティー」ウィンドウには、現在ご使用の特定の構成で設定可能なプロパティーが表示されます。

ただし**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、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ー固有プロパティー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、101 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 Express は、Windows 環境でのみ実行できます。UNIX 環境でコネクターを実行する場合は、Windows で Connector Configurator Express 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 Express の始動

Connector Configurator Express は、以下の 2 種類のモードで始動し、実行することができます。

- 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 Express の実行

Connector Configurator Express をブローカーと連携させずに別個に実行して、コネクター構成ファイルを編集することができます。

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 Business Integration Server Express」>「Toolset Express」>「開発」>「Connector Configurator Express」**をクリックします。
- **「ファイル」>「新規」>「構成ファイル」**を選択します。

Connector Configurator Express 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方法が便利です (106 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator Express の実行

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

Connector Configurator Express 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、**「コネクター」**を強調表示します。
3. System Manager メニュー・バーから、**「ツール」>「Connector Configurator Express」**をクリックします。「Connector Configurator Express」ウィンドウが開き、**「新規コネクター」**ダイアログ・ボックスが表示されます。

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。

1. 「System Manager」ウィンドウの「コネクター」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。

2. 「標準のプロパティー」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ーを確認します。

コネクター固有のプロパティー・テンプレートの作成

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

コネクター固有プロパティーのテンプレートを新規に作成するか、または既存のファイル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、101 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、新しい名前でこのテンプレートを保管します。

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ーを作成し、プロパティーの一般特性および値を定義し、プロパティー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、新規コネクター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。

テンプレートは以下のように作成します。

1. **「ファイル」>「新規」>「コネクター固有プロパティー・テンプレート」**をクリックします。
2. 以下のフィールドを含む**「コネクター固有プロパティー・テンプレート」**ダイアログ・ボックスが表示されます。
   - **「テンプレート」**、**「名前」**

     このテンプレートが使用されるコネクター (またはコネクターのタイプ) を表す固有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。
   - **「旧テンプレート」**、**「変更する既存のテンプレートを選択してください」**

     「テンプレート名」表示に、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。
   - テンプレートに含まれているコネクター固有のプロパティー定義を調べるには、**「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ー定義のリストが**「テンプレートのプレビュー」**表示に表示されます。テンプレートを作成するときには、ご使用のコネクターに必要なプロパティー定義に類似したプロパティー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。
3. **「テンプレート名」**表示からテンプレートを選択し、その名前を**「名前の検索」**フィールドに入力し (または**「テンプレート名」**で自分の選択項目を強調表示し)、**「次へ」**をクリックします。

ご使用のコネクターで使用するコネクター固有のプロパティー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。

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、**「プロパティー: コネクター固有プロパティー・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ーの「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:**
  プロパティー・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

プロパティー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、プロパティーの最大長、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. **「プロパティーを編集」**表示でプロパティーの名前を選択します。
3. **「最大長」**および**「最大複数値」**のフィールドで、変更を行います。次のステップで説明するように、プロパティーの**「プロパティー値」**ダイアログ・ボックスも開かない限り、そのプロパティーの変更内容は受け入れられません。
4. 値テーブルの左上の隅にあるボックスを右マウス・ボタンでクリックしてから、**「追加」**をクリックします。**「プロパティー値」**ダイアログ・ボックスが表示されます。このダイアログ・ボックスではプロパティーのタイプに応じて、値だけを入力できる場合と、値と範囲の両方を入力できる場合があります。適切な値または範囲を入力し、**「OK」**をクリックします。
5. **「値」**パネルが最新表示され、**「最大長」**および**「最大複数値」**で行った変更が表示されます。以下のような 3 つの列があるテーブルが表示されます。

   「値」の列には、**「プロパティー値」**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、**「プロパティー値」**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ー・テンプレート」**ダイアログ・ボックスが表示されます。

依存プロパティーは、別のプロパティー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ーです。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。プロパティーを依存プロパティーとして指定し、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ー」**表示で、依存プロパティーとして指定するプロパティーを選択します。
2. **「プロパティーを選択」**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ーを選択します。
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、依存プロパティー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ー」**表示で依存プロパティーを強調表示させて矢印をクリックし、**「依存プロパティー」**表示に移動させます。
6. **「完了」**をクリックします。入力した情報が、Connector Configurator Express によって、Connector Configurator Express 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 XML 文書として保管されます。

新しい構成ファイルを作成

コネクター構成ファイルを作成するには、コネクター固有のテンプレートから作成するか、既存の構成ファイルを変更します。

コネクター固有のテンプレートからの構成ファイルの作成

コネクター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」>「コネクター構成」**をクリックします。
2. 以下のフィールドを含む**「新規コネクター」**ダイアログ・ボックス表示されます。
   - **名前**

コネクター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ー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 Express 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**

  デフォルトのブローカーは ICS です。この値は変更できません。

- **「コネクター固有プロパティー・テンプレート」を選択します。**

  ご使用のコネクター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、**「プロパティー・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ー固有のプロパティーが表示されます。

  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ー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ー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・コネクターを保管」**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 Express 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、コネクターの始動ファイルで指定するコネクター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 Express」ウィンドウの各タブにあるフィールドに値を入力し、コネクター定義を完了します。

既存ファイルの使用

既存ファイルを使用してコネクターを構成するには、Connector Configurator Express でそのファイルを開き、構成を修正してから、構成ファイル (`*.cfg`) として保管する必要があります。

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。

- コネクター定義ファイル。
  コネクター定義ファイルは、特定のコネクターのプロパティーと、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。コネクターの配布パッケー

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 コネクターの場合は CN_XML.txt です)。

- InterChange Server Express リポジトリー・ファイル。
  以前にコネクターの InterChange Server Express インプリメンテーションの際に使用された定義が、そのコネクターの構成に使用されたリポジトリー・ファイルに残されていることがあります。そのようなファイルの拡張子は、通常 .in または .out です。
- コネクターの以前の構成ファイル。
  こ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、コネクターのコネクター固有プロパティーのほとんど、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、コネクター構成ファイルは、ファイルを開いて、プロパティー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ーを構成するには、Connector Configurator Express でそのファイルを開き、構成を修正してから、再度保管する必要があります。

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

1. Connector Configurator Express 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・コネクターを開く」**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)
   - InterChange Server Express リポジトリー (*.in、*.out) (InterChange Server Express Repository (*.in, *.out))

     これまでリポジトリー・ファイルを使用してコネクターを構成していた場合は、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ー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 (*.*)

     コネクターのアダプター・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、適切なコネクター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ー構成を開くには、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 Express 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 Express」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。

Connector Configurator Express では、以下のセクションに記載されているプロパティーの値を設定する必要があります。

- 107 ページの『標準コネクター・プロパティーの設定』
- 107 ページの『アプリケーション固有の構成プロパティーの設定』
- 108 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』
- 110 ページの『関連付けられたマップ』
- 111 ページの『トレース/ログ・ファイル値の設定』

注: コネクターが JMS メッセージングを使用するものである場合、データをビジネス・オブジェクトに変換するデータ・ハンドラーを構成できるように、追加のカテゴリーが表示されることがあります。詳細については、 112 ページの『データ・ハンドラー』を参照してください。

構成ファイル・プロパティーの設定

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、または既存のコネクター構成ファイルを開くと、Connector Configurator Express に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。

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

- コネクターの標準プロパティーは、コネクター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用されます。すべてのコネクターが同じ標準プロパティーのセットを使用します。これらのプロパティーの説明は、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ーは、コネクター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。各コネクターには、そのコネクター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ーがあります。これらのプロパティーには、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ーおよび推奨値が記述されています。

「標準プロパティー」と**「コネクター固有プロパティー」**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、標準のプロパティーを表します。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ー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ーを表します。これらのプロパティーは、アプリケーションまたはコネクター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ーの除去も可能です。
- 「値」フィールドは構成可能です。
- 各プロパティーごとに表示される **「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。

標準コネクター・プロパティーの設定

標準のプロパティー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. 標準のプロパティー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 Express 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 Express 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。**「標準のプロパティー」**(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ーの設定

アプリケーション固有の構成プロパティーの場合、プロパティー名の追加または変更、値の構成、プロパティーの削除、およびプロパティーの暗号化が可能です。プロパティー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。プロパティーを追加するときは**「追加」**をクリックします。子プロパティーを追加するには、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. プロパティーまたは子プロパティーの値を入力します。
3. プロパティーを暗号化するには、**「暗号化」**ボックスを選択します。
4. 『標準コネクター・プロパティーの設定』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。

各プロパティーごとに表示される「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ー・プロパティー名を変更すると、コネクターに障害が発生する可能性があります。コネクター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、特定のプロパティー名が必要である場合があります。

コネクター・プロパティーの暗号化

「プロパティーを編集」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ー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。

各プロパティー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。

プロパティーに複数の値がある場合には、プロパティーの最初の値に**「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、そのプロパティーのすべての値が暗号化されます。プロパティーの複数の値を暗号化解除するには、そのプロパティーの最初の値の**「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、すべての複数値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ー*』の 83 ページの『プロパティー値の設定と更新』にある更新メソッドの説明を参照してください。

コネクター・プロパティーはほとんどが静的なプロパティーであり、それらの**更新メソッド**は**コンポーネント再始動**です。変更を有効にするには、変更したコネクター構成ファイルを保管した後、コネクターを再始動する必要があります。

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 Express 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。

サポートされるビジネス・オブジェクトを指定するときには、指定するビジネス・オブジェクトとそのオブジェクトに対応するマップが、システムに存在していなければなりません。ビジネス・オブジェクト定義 (データ・ハンドラー・メタオブジェクトのビジネス・オブジェクト定義を含みます) とマップ定義は、統合コンポーネント・ライブラリー (ICL) プロジェクトに保管されている必要があります。ICL プロジェクトの詳細については、「*WebSphere Business Integration Server Express ユーザーズ・ガイド*」を参照してください。

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。詳細につ

いては、本書のビジネス・オブジェクトに関する章と、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

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 Express」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 Express」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、コネクター定義が変更され、削除されたビジネス・オブジェクトはコネクターのこの実装で使用不可になります。コネクター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、コネクター・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、コネクターのアプリケーション固有ビジネス・オブジェクトは、そのコネクター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ー・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 Express」ウィンドウでは、「エージェント・サポート」を選択しても問題ないかどうかの検証は行われません。

最大トランザクション・レベル

コネクターの最大トランザクション・レベルは、そのコネクター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ーの場合、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。

関連付けられたマップ

各コネクターは、ビジネス・オブジェクト定義とそれらに関連付けられたマップのうち現在 InterChange Server Express でアクティブであるものを示す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ライブ・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に、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 Express」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。InterChange Server Express は、ブート時、各コネクターのサポートされるビジネス・オブジェクトのそれぞれ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 つのマップを見付けて、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 Express」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 InterChange Server Express 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

「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。

トレース/ログ・ファイル値の設定

コネクター構成ファイルまたはコネクター定義ファイルを開くと、Connector Configurator Express は、そのファイルに含まれるロギングとトレースに関する値をデフォルト値として使用します。これらの値は、Connector Configurator Express 内で変更できます。

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ーの**「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、指定する格納場所に移動し、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。このタブは、アダプターが保証付きイベント・デリバリーを利用するものである場合に使用可能になります。

これらのプロパティーに使用する値については、標準プロパティーに関する付録の『ContainerManagedEvents』の説明を参照してください。

構成ファイルの保管

構成ファイルの作成とそのファイルに含まれるプロパティーの設定が完了したら、使用するコネクターに応じた適切な場所にそのファイルを配置する必要があります。ICL プロジェクトに構成を保管し、保管されたファイルを System Manager から InterChange Server Express へロードしてください。

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。

System Manager でのプロジェクトの使用方法と、配置の詳細については、「*User Guide for IBM WebSphere Business Integration Server Express*」を参照してください。

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、コネクターが使用する始動ファイルを開き、コネクター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ー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 Express の使用

Connector Configurator Express はグローバル化されており、構成ファイルと統合ブローカーの間での文字変換を処理できます。Connector Configurator Express 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 Express 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ーのドロップ・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、製品ディレクトリーの ¥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 プロパティー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>
```

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で提供し、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すが、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が、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証できません。 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にお願いします。

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、単に目標を示しているものです。

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 および Pentium 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。
System Manager には、Eclipse Project (http://www.eclipse.org/) により開発されたソフトウェアが含まれています。

WebSphere Business Integration Server Express V4.3 および WebSphere Business Integration Server Express Plus V4.3

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12