IBM WebSphere Business Integration
Adapters

Adapter for Portal Infranet ユーザーズ・ガイド

バージョン 4.2.x

IBM WebSphere Business Integration
Adapters

Adapter for Portal Infranet ユーザーズ・ガイド

バージョン 4.2.x

お願い

本書および本書で紹介する製品をご使用になる前に、113 ページの『付録 C. 特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere InterChange Server バージョン 4.2、WebSphere Business Integration Adapters バージョン 2.3.0、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: IBM WebSphere Business Integration Adapters
Adapter for Portal Infranet User Guide
Version 4.2.x

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 2004.1

目次

本書について

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

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

対象読者

本書は、WebSphere Business Integration システムの一部としてコネクターを実装する WebSphere コンサルタントおよびお客様を対象読者としています。本書に記載されている情報を使用するには、以下の領域に関する知識が必要です。

- コネクターの開発
- ビジネス・オブジェクトの開発
- Portal Infranet アプリケーションのアーキテクチャー

関連資料

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

以下のサイトから、関連資料をインストールすることができます。

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

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

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

http://www.ibm.com/websphere/integration/wicserver/infocenter
http://www.ibm.com/websphere/integration/wbicollaborations/infocenter

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

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

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

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

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

表記上の規則

本書は下記の規則に従って編集されています。

`Courier フォント`	コマンド名、ファイル名、入力情報、システムが画面に出力した情報など、記述されたとおりの値を表します。
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。
青のアウトライン	青のアウトラインは、マニュアルをオンラインで表示するときにのみ見られるもので、相互参照用のハイパーリンクを示します。 アウトラインの内側をクリックすることにより、参照先オブジェクトにジャンプできます。
{ }	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。
< >	命名規則により、1 つの名前の個々の要素を互いに区別するために、不等号括弧によって個々の要素が囲まれます。例えば、`<server_name><`*`connector_name`*`>tmp.log` のように使用します。
/, \	本書では、ディレクトリー・パスに円記号 (¥) を使用します。UNIX システムの場合には、円記号 (¥) はスラッシュ (/) に置き換えてください。すべての WebSphere Business Integration システム製品のパス名は、ご使用のシステムで製品がインストールされたディレクトリーを基準とした相対パス名です。
%text% および *$text*	パーセント (%) 記号で囲まれた文字列は、Windows `text` のシステム変数またはユーザー変数を示します。UNIX 環境での同等表記は、*$text*です。これは、*text* UNIX 環境変数の値を示します。
ProductDir	製品がインストールされているディレクトリーを表します。

本リリースの新機能

バージョン 4.2.x

Adapter for Portal Infranet が HP-UX 11i 上でサポートされるようになりました。

4.2x バージョンから、Adapter for Portal Infranet は Microsoft Windows NT 上ではサポートされなくなりました。

アダプターのインストール情報は、本書から移動しました。この情報の新たな入手先については、第 2 章、 14 ページの『Portal Infranet アダプターおよびその他のファイルのインストール』を参照してください。

以前のリリース

以前のリリースでの新機能

バージョン 4.1.x

カスタム・ビジネス・オブジェクトの開発モデルを提供するため、 58 ページの『Portal Infranet ビジネス・オブジェクト定義の完全サンプル』が追加されました。

アダプターは、WebSphere Application Server を統合ブローカーとして使用できるようになりました。詳細については、 12 ページの『互換性』を参照してください。
アダプターは、以下のプラットフォーム上で実行されます。

- Solaris 7、8
- AIX 5.x

バージョン 4.0.x

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

コネクターは、CWSAPGEN ユーティリティーを PORTALODA に置き換えています。詳細については、 61 ページの『第 4 章 PortalODA 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。

バージョン 3.1.x

国際化対応コネクターは、IBM WebSphere Business Integration Adapter for Portal Infranet とともに提供されます。

バージョン 3.0.x

コネクターのこのリリースには、以下の新しい機能が含まれています。コネクターは国際化に対応しています。詳細については、 9 ページの『ロケール依存データの処理』および 77 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。コネクターは、AIX 4.3.3 上で以下のソフトウェアをサポートします。

- Oracle 8.1.7 および Portal Infranet 6.2 SP1
- DB2 7.1.0 および Portal Infranet 6.2 SP1

バージョン 2.5.x

IBM WebSphere Business Integration Adapter for Portal Infranet には、Portal Infranet のコネクターが含まれています。このアダプターは、InterChange Server (ICS) および WebSphere MQ Integrator 統合ブローカーとともに動作します。統合ブローカーとは、異種のアプリケーション・セット間の統合を実行するアプリケーションです。統合ブローカーは、データ・ルーティングなどのサービスを提供します。アダプターには、以下のエレメントが含まれます。

- Portal Infranet 固有のアプリケーション・コンポーネント
- ビジネス・オブジェクトの例
- IBM WebSphere Adapter フレームワーク。構成エレメントは以下のとおりです。
  - 開発ツール (Business Object Designer と Connector Configurator を含む)
  - API (ODK、JCDK、および CDK を含む)

本書には、このアダプターを InterChange Server (ICS) および WebSphere MQ Integrator の両方の統合ブローカーとともに使用する方法についての情報が記載されています。

注: コネクターは、このリリースから Portal Infranet 6.2.0 をサポートします。

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

バージョン 2.4.x

このコネクターのバージョン 2.4.x に加えられた変更は、本書の内容には記載されていません。

バージョン 2.3.x

欠陥を修正し IBM CrossWorlds インフラストラクチャー・バージョン 4.0.0 との互換性を保証するために、小規模の変更が加えられています。

バージョン 2.2.x

Portal Infranet 6.1.0 をサポートするようになりました。

バージョン 2.1.x

- Portal Infranet 6.0.1 をサポートするようになりました。
- コネクターは、UNIX システム上にインストールして実行することができます。
- 本書は、大幅に再編成および改訂されました。

第 1 章 コネクターの概要

この章では、IBM WebSphere Business Integration Adapter for Portal Infranet のコネクター・コンポーネントについて概説します。以下のセクションから構成されています。

コネクターは、コネクター・フレームワークとアプリケーション固有コンポーネントから構成されています。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なので、コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントとの仲介役の機能を果たします。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合わせたコードが格納されています。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントとの間で以下のようなサービスを提供します。

- ビジネス・オブジェクトの送信および受信
- 始動メッセージや管理メッセージの交換の管理

本書では、コネクター・フレームワークと、コネクターと呼ぶアプリケーション固有コンポーネントについて説明しています。

このコネクターを使用すると、WebSphere MQ Integrator Broker または IBM WebSphere Interchange Server (ICS) は、ビジネス・オブジェクトをやり取りすることによって Portal Infranet と通信できます。Infranet アプリケーションは、顧客アカウント管理のために Portal Infranet ソフトウェアによって開発されたソフトウェア・プログラムの集合です。アカウント番号や請求情報などの顧客情報が、Infranet データベースに保管されます。

コネクターと Portal Infranet アプリケーションは、Infranet のソケット・ベースの API を使用して通信を行います。コネクターは Infranet API によって提供される関数を使用してトランザクションを処理し、Infranet アプリケーションは変化が生じるとイベント・モジュールを介して統合ブローカー (WebSphere MQ Integrator Broker または ICS) に通知します。

コネクターのコンポーネント

Portal Infranet のコネクターには以下のコンポーネントが含まれます。

- コネクター: ビジネス・オブジェクト動詞のサポートとイベント・ポーリング機構を実装した Java .jar ファイル。
- WebSphere Business Integration Adapter イベント機能モジュール: Infranet アプリケーションにおけるイベント通知機構を実装した、Windows 上の C++ DLL と UNIX 上の SO ファイル。このモジュールは、統合ブローカーに関係のある Infranet イベントを選択して、それらを Infranet データベースの表に保管します。コネクターはこの表を定期的にポーリングします。

コネクターはビジネス・オブジェクトを生成し、統合ブローカーに送ります。また、統合ブローカーからのビジネス・オブジェクト要求にも対応します。コネクターは、ロギング・メッセージとトレース・メッセージを生成し、そのメッセージをファイルまたはコネクター・コンソールに出力したり、統合ブローカーに送信したりします。

図 1 は、コネクターのアーキテクチャーと Infranet アプリケーションにおけるコネクターのイベント機構を図示したものです。

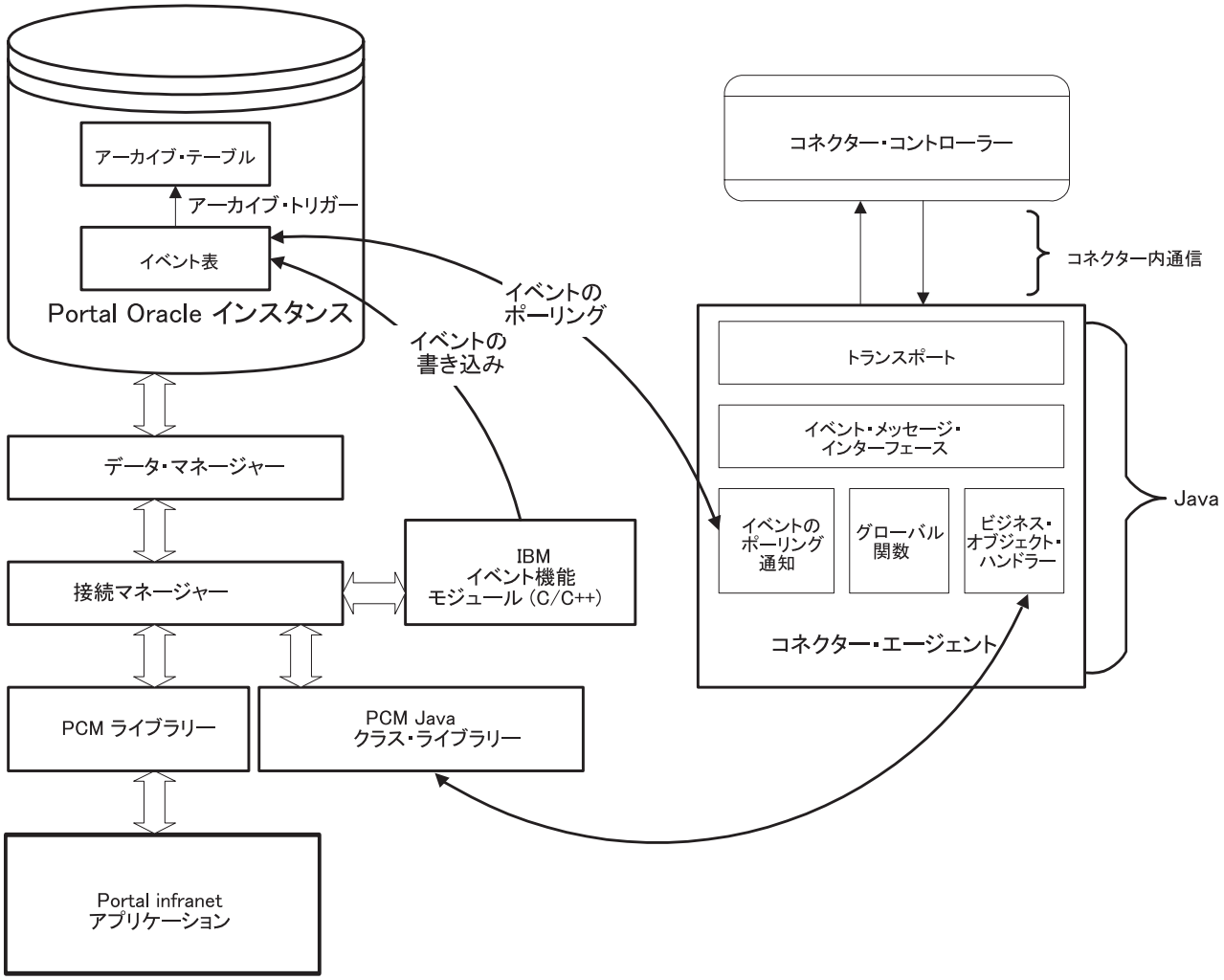


図 1. コネクター・アーキテクチャー

コネクターは、Portal Infranet 接続マネージャーと対話するための API として Portal Communications Module (PCM) Java クラス・ライブラリーを使用します。このアーキテクチャーの利点は、Windows と UNIX の両方について Portal Infranet が Java 仮想マシン上の PCM Java クラス・ライブラリーをサポートしていることです。これにより、コネクターは両方のプラットフォームで実行可能となります。この API は、コネクター内のビジネス・オブジェクト・ハンドラーによって統合ブローカーと Portal の間の情報交換のために使用されます。WebSphere Business Integration Adapter イベント機能モジュールは C++ PCM ライブラリーを使用します。

コネクターの動作

以下のセクションでは、コネクターがビジネス・オブジェクト要求を処理する方法、およびイベント通知を処理する方法を説明します。

メタデータ主導型コネクターの動作

コネクターはメタデータ主導型です。コネクターは、ビジネス・オブジェクトのタイプや含まれる変数にかかわらず、すべてのビジネス・オブジェクトの検索と実行依頼を処理するように設計されています。コネクターがメタデータ主導型となるためには、Portal Infranet 用ビジネス・オブジェクトが以下の情報を含む必要があります。

- Infranet のデータ・ディクショナリーに示されている各属性のフィールド名。このフィールド名には、API 資料に記載されている各ピン・フィールド名のピン・フィールド番号が含まれます。フィールド名はアプリケーション固有情報として属性レベルで指定され、データ・ディクショナリーを使用してコネクターによって番号に変換されます。
- このビジネス・オブジェクトによりサポートされる命令コード。命令コードは、ビジネス・オブジェクトの動詞レベルで指定されます。Infranet 命令コードは、クライアント・アプリケーションおよびスクリプトによって、顧客関連情報の管理、オンライン・アカウントの作成、顧客情報の収集と追跡、およびサード・パーティー・システムと Infranet との統合のために使用される操作です。

Portal 用ビジネス・オブジェクトのメタデータの詳細については、35 ページの『第 3 章 ビジネス・オブジェクトの理解』を参照してください。

ビジネス・オブジェクトの処理

コネクターが WebSphere Business Integration システムからビジネス・オブジェクト要求を受信すると、コネクターのビジネス・オブジェクト・ハンドラーがそのビジネス・オブジェクトを処理します。ビジネス・オブジェクト・ハンドラーは、アプリケーション固有のオブジェクトと Portal Infranet API とを結ぶ橋の役割をします。ビジネス・オブジェクト・ハンドラーは、API に対する Portal Infranet 操作の実行依頼、および Infranet イベントの結果として WebSphere Business Integration システムに送信されるアプリケーション固有のビジネス・オブジェクトの作成を担当します。ビジネス・オブジェクト・ハンドラーは、ビジネス・オブジェクト内のデータとあらゆるメタデータを使用して、格納可能オブジェクトの実行を Portal に依頼するために Infranet Java API を呼び出します。この操作が完了すると、統合ブローカーに状況が戻されます。

図 2 のフローチャートには、ビジネス・オブジェクト・ハンドラーがビジネス・オブジェクト要求を処理する方法が詳細に示されています。ビジネス・オブジェクト・ハンドラーは、ビジネス・オブジェクトから動詞およびキーの属性を抽出します。次に、動詞を使用して、ビジネス・オブジェクトの処理のために行う関数呼び出しを決定します。この例では、動詞が update 動詞である場合には UpdateObject 関数が呼び出されます。

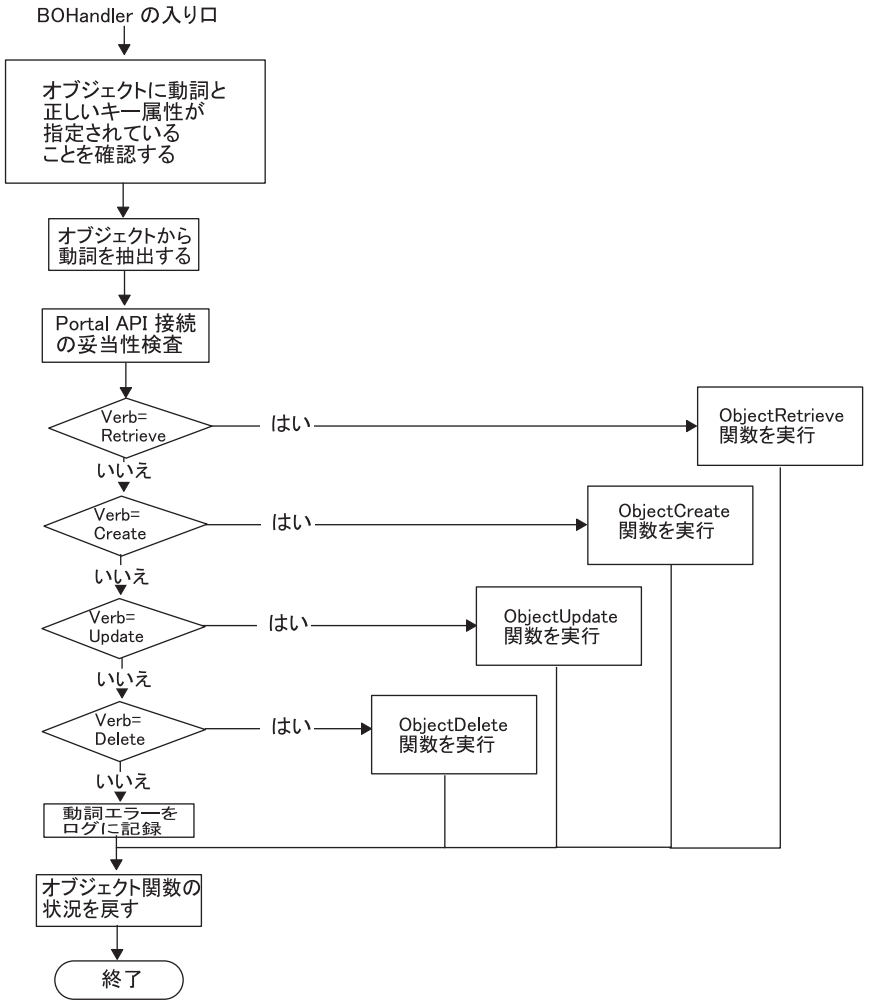


図 2. ビジネス・オブジェクト処理の全体図

Retrieve 動詞の処理

ビジネス・オブジェクト・ハンドラーの Retrieve メソッドは、Portal Infranet からオブジェクトを検索し、WebSphere Business Integration Adapter ビジネス・オブジェクトにアプリケーション情報を取り込みます。コネクターの Retrieve メソッドは以下の処理を行います。

1. Portal Infranet 接続が有効であるかどうか検査します。接続が無効な場合、再インスタンス化する必要があります。処理中に接続が切断した場合、コネクターは BON_FAIL 状況を戻して Infranet との接続における問題発生を示します。

2. オブジェクトのアプリケーション固有情報を検索します。動詞に対して指定されているアプリケーション固有情報から、Portal Infranet オブジェクトの検索のために呼び出す必要がある命令コードが得られます。
3. ビジネス・オブジェクトおよび属性に対するアプリケーション固有情報に基づいて命令コードのフィールド・リストを作成します。

   注: フィールド・リストはフィールドと値の組から構成される可変長リストです。フィールド・リスト は Infranet の命令コードおよび関数に入出力パラメーターを提供します。
4. フィールド・リストを入力として、さらに空のフィールド・リストを出力先として指定された命令コードを呼び出します。
5. 直前のステップが成功の場合、戻されるフィールド・リスト構造には、WebSphere Business Integration Adapter ビジネス・オブジェクトにより定義された格納可能オブジェクトに対応する、完全に情報が取り込まれたフィールド・リスト・オブジェクトが含まれます。フィールド・リストは WebSphere Business Integration Adapter ビジネス・オブジェクトと 1 対 1 で対応するので、フィールド・リストのフィールド名/型を示す属性レベルのアプリケーション固有情報に基づき WebSphere Business Integration Adapter ビジネス・オブジェクトの全属性を検索するためにトラバースされます。
6. ビジネス・オブジェクトに情報を取り込んで統合ブローカーに送信します。

図 3 に Retrieve メソッドの機能を示します。Retrieve メソッドは、ある属性に対して実行すべき処理を属性タイプに応じて決定します。基本的な属性タイプ (ストリングなど) の場合、ビジネス・オブジェクト・ハンドラーはフィールドへの情報の取り込みを動的に行います。Retrieve メソッドの実行中に子ビジネス・オブジェクトが出現した場合、メソッドはオブジェクトの中を下に移動してその子ビジネス・オブジェクトの基本属性の位置まで到達します。その後 Retrieve メソッドは子オブジェクトの全基本属性の間を循環し、親オブジェクトの基本属性を引き継ぎます。

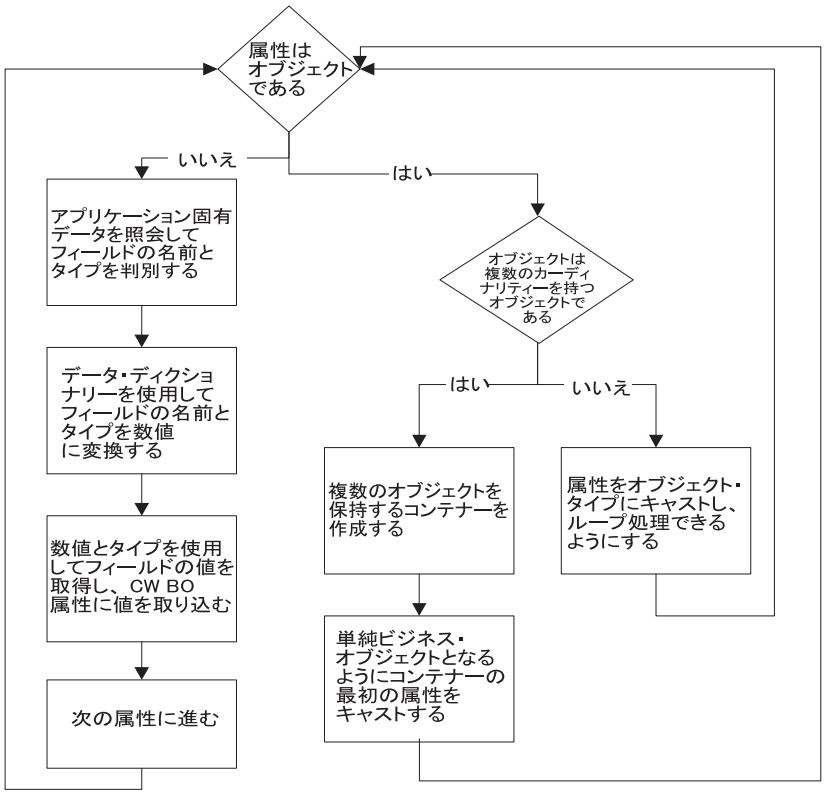


図 3. Retrieve 動詞の処理フローチャート

Create 動詞および Update 動詞の処理

Create 動詞または Update 動詞を処理するために、ビジネス・オブジェクト・ハンドラーは Infranet API オブジェクト (フィールド・リスト) を構成し、それを Infranet API に渡します。ビジネス・オブジェクト・ハンドラーは以下のステップを実行します。

1. WebSphere Business Integration Adapter ビジネス・オブジェクトからアプリケーション固有のビジネス・オブジェクトを検索します。
2. Portal Infranet API オブジェクトに情報を取り込みます。フィールド・リストがインスタンス化されると、コネクターはオブジェクトの属性 1 つ 1 つにアクセスし、フィールド・リストに取り込む値を探し出します。このプロセスでは、オブジェクト内で複数の層をなす可能性がある属性を、オブジェクトの中から探し出す必要があります。
3. Portal Infranet 接続がまだ有効であるかどうか検査します。処理中に接続が切断した場合、コネクターは `BON_FAIL` 状況を戻して Infranet との接続における問題発生を示し、接続は再インスタンス化する必要があります。
4. ビジネス・オブジェクト動詞のアプリケーション固有情報を使用して、Create 操作または Update 操作のために適切な命令コードを動的に呼び出します。パラメーターが集められて配列の中に置かれ、機能ストリングがアセンブルされると、適切な命令コードを含むコンテキスト関数が呼び出されます。

5. 独自の命令コードを持つ子ビジネス・オブジェクトを処理するために、ビジネス・オブジェクト・ハンドラーは子ごとに別々のフィールド・リストを取り込んでから適切な命令コードを呼び出します。
6. 操作が完了したら状況を戻します。

イベント通知

Infranet には、アプリケーション内で発生するアクションの追跡を可能にするイベント機構が備わっています。Infranet 内でユーザーがあるアクションを実行すると、アプリケーションは関連するイベントを生成します。

WebSphere Business Integration Adapter イベント・モジュールはそのイベントをチェックし、コネクターに関係があるかどうか判断します。関係する場合、イベント・モジュールはそのイベントの WebSphere Business Integration Adapter イベント表にエントリーを生成します。

Infranet におけるイベント検出

Infranet におけるイベント検出は、イベント通知機構によって呼び出されるカスタム Infranet 機能モジュールによって実装されます。この機能モジュールは統合ブローカーによって提供され、2 つの構成ファイルと共に動作して Infranet イベントを識別し、WebSphere Business Integration Adapter イベント表にイベントを書き込みます。

Infranet オブジェクトに変更点が生じると、永続イベントが発生します。あるイベントの発生時に特定の命令コードを呼び出すように Infranet を構成できます。このため、統合ブローカーの提供する構成フラット・ファイルは、Portal 用ビジネス・オブジェクトに関連するイベントに応じて WebSphere Business Integration Adapter イベント機能モジュールを呼び出すように Infranet を構成します。この構成ファイルは「pin_notify_cw」と呼ばれ、Infranet と共に提供される `load_pin_notify` ユーティリティーを使用して Infranet にロードされます。

イベント・モジュールはあるイベントを受け取ると、そのイベント・オブジェクトから情報を抽出し、WebSphere Business Integration Adapter イベント表に新しいエントリーを作成します。Infranet 内のイベントは実際には Infranet 格納可能クラスのインスタンスであり、作成、変更、削除の各イベントは特定の Infranet 格納可能クラスに関連しています。例えば、あるユーザーが顧客に関連する特定の連絡先を変更する場合、Infranet は格納可能クラス `/event/customer/nameinfo` のインスタンスを生成します。

イベント・モジュールは所有するイベント・モジュール構成ファイルを使用して、発生したイベントの判別、格納可能クラス (および関連ビジネス・オブジェクト) の変更された部分の確認、および発生したアクションのタイプの判別を行います。イベント・モジュールは構成ファイル `event_code.txt` を使用して Infranet イベントを調査し、WebSphere Business Integration Adapter イベント表にそのイベントを反映するレコードを取り込みます。

コネクター用のイベント通知機構は、Infranet により使用される Oracle データベース・インスタンスの内部に作成された次の 3 つの表を使用します。

- XWORLDS_Events: すべての保留イベントを格納します。
- XWORLDS_Archive_Events: コネクターが処理したイベントを保存します。

- XWORLDS_Current_Event_ID: 最終イベントの ID 番号を格納します。

1 つ目の表のスキーマは、コネクターに関係する Infranet により送信されるイベントごとに記録される情報を指定します。この表のレイアウトはアーカイブ表にも使用されます。

イベント検出とその関連処理は、Infranet トランザクション内で実行されます。Infranet はトランザクション内のカスタム・プロセスを呼び出して、処理の結果を待ちます。カスタム・プロセスがエラーを戻した場合、そのトランザクションは打ち切られます。この仕組みにより、コネクターがイベントを失うことはありません。

注:

既知の問題: イベント通知モジュールは、自身に送られてきたすべてのポータル・イベントのユーザー ID を確認します (pin_notify_cw ファイルで定義)。モジュールに送られてきたイベントに対応する PIN_FLD_USERID がない場合はエラーとなり、オブジェクトのオンライン保存時に問題が発生します。このようなタイプのイベントは、FList または格納可能クラスを使用して、正しい ID を組み込むための調整を行う必要があります。crossworlds.cnf 構成ファイルで定義されたログ・ファイルにこれらのエラーがないか確認してください。

イベント・モジュールは、ユーザー ID を調べることにより、コネクターによってアプリケーションに送信されたイベントがイベント・キューに追加されるのを防ぎます。これを「ピンポン」と呼びます。

/event/customer/billinfo は、この種の問題のあるイベント・タイプです。

イベントの検索

コネクターは、Infranet データベース・インスタンス内に設定された XWORLDS_Events 表をポーリングすることによりイベントの有無を確かめます。コネクターは SQL SELECT ステートメントを使用してポーリングを行い、XWORLDS_Events 表からエントリーを抽出します。選択されるイベントの数は、コネクターの PollQuantity プロパティーによって指定されます。

ポーリングは、コネクター内の `pollForEvents()` メソッドで行われます。コネクターは、WebSphere Business Integration Adapter コネクター・プロパティーに設定された PollFrequency の頻度でイベント表をポーリングします。表から新しい行が検出されると、イベント・データが検索されて、コネクターはそのイベントを以下のように処理します。

1. ポーリング関数は空のビジネス・オブジェクトを作成し、動詞を Retrieve に設定して、イベント・レコードを使ってキーを設定します。ビジネス・オブジェクトはコネクターのビジネス・オブジェクト・ハンドラーに送信されます。
2. ビジネス・オブジェクト・ハンドラーはイベント・データを使用して Infranet Java API を呼び出し、Portal Infranet の格納可能オブジェクトを検索します。
3. ビジネス・オブジェクト・ハンドラーは、格納可能オブジェクトを WebSphere Business Integration Adapter 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、動詞をイベント・レコード内のアクションに設定してから、ビジネス・オブジェクトを統合ブローカーに送信します。

ビジネス・オブジェクトが WebSphere Business Integration システムに送信されると、イベント表のエントリーは XWORLDS_Archive_Events 表に保存され、イベント表からは削除されます。

ポーリング・メソッドが呼び出される時間間隔は、コネクター・プロパティーの PollFrequency を変更することによって調整できます。このプロパティーを設定するには、統合ブローカーを使用します。

Infranet アプリケーションへの接続

API を使用して Portal Infranet 接続マネージャーに接続する場合、コネクターは以下の処理を行います。

1. Portal Infranet コンテキストの新しいインスタンスを作成します。
2. 命令コードを実行する場合は、コネクターはプールからのコンテキストを使用し、そのコンテキストを使用中プールに追加します。
3. タスクが完了すると、コンテキストは空きプールに戻されます。

接続ステートメントは、リポジトリーに定義されているコネクターのアプリケーション固有プロパティーの値を使用します。

プールのコンテキスト・インスタンスは、コネクターが終了するとクローズされます。

ロケール依存データの処理

コネクターは、2 バイト文字セットをサポートし、指定の言語でメッセージ・テキストを配信できるように国際化対応されています。コネクターが、1 文字コード・セットを使用する地域から異なるコード・セットを使用する地域にデータを転送する場合は、文字変換を実行し、データの意味を維持します。Java 仮想マシン (JVM) 内の Java ランタイム環境では、データが Unicode 文字コード・セットで表現されます。Unicode には、既知の文字コード・セット (単一バイトとマルチバイトの両方) における文字のエンコードが、組み込まれています。IBM CrossWorlds システムでは、ほとんどのコンポーネントが、Java で書かれています。したがって、普通、IBM CrossWorlds コンポーネント間のデータ転送の場合には、文字変換の必要はありません。エラーおよび通知メッセージを該当する言語で、該当する国または地域のためにログに記録するには、実際の環境に対して Locale 標準構成プロパティーを構成します。これらのプロパティーの詳細については、77 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。

第 2 章 コネクターのインストールと構成

この章では、コネクターをインストール、構成する方法について説明します。以下の項目から構成されています。

IBM WebSphere Business Integration Adapter for Portal Infranet のコネクター・コンポーネントには、インストールおよび構成を必要とする 2 つのコンポーネントがあります。

- コネクター: これは、Java `.jar` ファイルです。コネクターは、コネクターの動詞のサポートとイベント・ポーリング機構を実装します。
- WebSphere Business Integration Adapter イベント機能モジュール: イベント機能モジュールは、イベント通知を実装した実行可能モジュールです。このモジュールは、統合ブローカーに関係のある Infranet イベントを選択して、それらをデータベース表に保管します。

この章では、コネクターのコンポーネントのインストール方法および構成方法と、Portal Infranet アプリケーションをコネクターとともに動作させるための構成方法を説明します。

注: 本書では、いくつかのコード・ファイルの例を除き、ディレクトリー・パスの規則として円記号 (¥) を使用します。UNIX システムの場合には、円記号 (¥) はスラッシュ (/) に置き換えてください。特に注意がない限り、すべてのファイルのパス名は、使用システムで WebSphere Business Integration Adapter 製品がインストールされたディレクトリーを基準とした相対パス名です。

互換性

アダプターが使用するアダプター・フレームワークは、アダプターと通信する統合ブローカーのバージョンとの互換性を備えている必要があります。Adapter for Portal Infranet のバージョン 4.2 は、以下のバージョンのアダプター・フレームワークおよび以下の統合ブローカーでサポートされています。

アダプター・フレームワーク: WebSphere Business Integration Adapter Framework バージョン 2.1、2.2、2.3.x、および 2.4。

統合ブローカー:

- WebSphere InterChange Server、バージョン 4.2.x
- WebSphere MQ Integrator バージョン 2.1.0
- WebSphere MQ Integrator Broker バージョン 2.1.0
- WebSphere Business Integration Message Broker、バージョン 5.0
- WebSphere Application Server Enterprise、バージョン 5.0.2 (WebSphere Studio Application Developer Integration Edition バージョン 5.0.1 と併用)

例外については、「リリース情報」を参照してください。

注: 統合ブローカーのインストール手順およびその前提条件については、次の資料を参照してください。WebSphere InterChange Server (ICS) については、「システム・インストール・ガイド (UNIX 版)」または「システム・インストール・ガイド (Windows 版)」を参照してください。

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

WebSphere Application Server については、「アダプター実装ガイド (WebSphere Application Server)」および次の資料を参照してください。
http://www.ibm.com/software/webservers/appserv/library.html

前提ソフトウェア

コネクターには以下の前提ソフトウェアが必要です。

- Java Portal Communications Module (PCM) Library `pcm.jar` および `pcmext.jar`
- WebSphere Business Integration Adapter に必要な一般的なサード・パーティー・ソフトウェアすべて。WebLogic Oracle JDBC ドライバーおよび JDBC ライブラリーなど。

イベント・モジュールには以下の前提ソフトウェアおよび設定が必要です。

- インストール済み Infranet アプリケーションと、関連する Oracle データベース。

- 環境変数 ORACLE_SID のデフォルト値に、Portal Infranet データベースの別名をセットする必要があります。

Windows システムにインストールする場合、イベント・モジュールには以下の前提ソフトウェアも必要になります。

- Portal Communications Module (PCM) Library pcm.dll
- WebSphere Business Integration Adapter for Portal Infranet のコネクター・コンポーネントとともに提供される Microsoft MFC と付属品 (MFC42.dll)。システムのパスにこのファイルがまだ存在しない場合は、このファイルを PATH システム変数で定義されているディレクトリーに配置する必要があります。このファイルを *ProductDir*¥bin ディレクトリーに配置すれば、その条件を満たせます。

Infranet アプリケーションの構成

コネクターに使用される Infranet アプリケーションを設定するためには、コネクター用のユーザー・アカウントを定義して、Infranet が使用する Oracle データベース内にイベント表とアーカイブ表を作成する必要があります。

Infranet アカウントの設定

Infranet Administrator を使用して、すべての権限を持つ Customer Service Representative (CSR) ユーザーを定義します。このユーザーはコネクターによって使用され、コネクターを識別します。このユーザー ID は、イベント・モジュール構成ファイル crossworlds.cnf とコネクター構成パラメーターに設定されます。Custom Event Facilities モジュールは、イベント挿入前にこの値をチェックして、コネクターによってアプリケーションに送信されたイベントがコネクターへ再送信されるのを防ぎます。このシナリオを「ピンポン」とも呼びます。

データベース内でのイベント表とアーカイブ表の作成

イベント表とアーカイブ表は、イベントをキューに入れてコネクターに取り出させる目的で使用されます。コネクター用のイベント通知機構では、Infranet によって使用される Oracle データベース・インスタンスの内部に 3 つのイベント表が作成される必要があります。必要な表は以下のとおりです。

- XWORLDS_Events: コネクターに関係するすべての保留 Infranet イベントが保管されるイベント表。
- XWORLDS_Archive_Events: コネクターに処理された後のイベントが書き込まれるアーカイブ表。
- XWORLDS_Current_Event_ID: 最終イベントの ID 番号を格納する表。

  注: この表を 0 に初期化する必要があります。

最初の 2 つの表は、コネクターに関係する Infranet イベントごとに記録される情報を指定します。アーカイブ表にはコネクターによって処理されたすべてのイベントが保管されます。

イベント表およびアーカイブ表を作成するには、*%ProductDir%*¥connectors¥Portal¥dependencies¥config_files に EventTable.sql ファイルをロードします。

イベント表およびアーカイブ表のスキーマの説明

イベント表には以下の列が含まれます。この表のレイアウトはアーカイブ表にも使用されます。

表 1. イベント表およびアーカイブ表のスキーマ

名前	タイプ	説明
Event_id	整数	イベントに応じた固有キー。キー値は XWORLDS_Current_Event_ID 表で生成されます。
Object_name	CHAR (80)	アプリケーション固有のビジネス・オブジェクト名。
Object_verb	CHAR (80)	イベントに関連付けられた動詞。
Object_key	VARCHAR	オブジェクトの基本キー (POID)。
Event_time	日時	イベントが発生した時刻。
Archive_time	日時	アーカイブ表のみ。イベントが Portal Infranet に受信された時刻。
Event_status	整数	イベントの状況: READY_FOR_POLL 0 SENT_TO_INTERCHANGE 1 UNSUBSCRIBED_EVENT 2 IN_PROGRESS 3 ERROR_PROCESSING_EVENT -1 ERROR_SENDING_EVENT_TO_INTERCHANGE -2
Event_comment	char 255	イベントに関する追加情報を提供するためのストリング。このコメントは、イベント・モジュール構成ファイル内に定義できます。
Event_priority	整数	イベントに関連付けられた優先順位。この数字が小さいほど、優先順位は高くなります。この優先順位は、イベント・モジュール構成ファイル内に定義できます。

Portal Infranet アダプターおよびその他のファイルのインストール

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

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

AIX/DB2 環境でのアダプターの構成

AIX システム上にアダプターを構成するには、以下の手順を実行してください。

DB2 ユーザーとして

1. Portal Infranet のインスタンスに使用する DB2 ユーザー ID、例えば、db2instl で、AIX システムにログインする。
2. 以下のコマンドを入力する。

   ```
   db2 catalog tcpip node local_DB2_hostname remote remote_DB2_hostname
   server DB2_instance_portnumber
   ```

   例:

   ```
   db2 catalog tcpip node db2host remote db2host server 50000
   ```

3. 以下のコマンドを入力する。

   ```
   db2 catalog database DB_name as DB_name at node DB2_hostname
   ```

例:

```
db2 catalog database CWPortal as CWPortal at node db2hos
```

4. ログオフする。

ユーザー「pin」として

1. pin として、AIX システムにログインする (必要であれば、pin ユーザー・アカウントを作成する)。
2. ファイル db2profile をユーザー pin のホーム・ディレクトリーにコピーした後、シェル・スクリプト・ファイル db2profile を実行するため、行 ../db2profile を .profile ファイルに追加してから、コネクターを始動する。

   このファイルは、DB2 用 Portal Infranet のインストールに使用したアカウントのホーム・ディレクトリー内のディレクトリー sqllib、例えば、/home/db2instl/sqllib に存在します。

   $COSSWORKDS/connectors/Portal/dependencies/config_files/DB2 から、ユーザー pin のホーム・ディレクトリーに、ファイル database.bnd をコピーする。
3. 以下のコマンドを入力する。

   db2 connect to *DB_name* user pin using pin

   例:

   ```
   db2 connect to CWPortal user pin using pin
   ```

4. 以下のコマンドを入力する。

   dbw bind database.bnd
5. 以下のコマンドを入力する。

   db2 disconnect all
6. ユーザー pin の .profile ファイルに、以下の行を追加する。

   export DB2DBDFT=*DB_name*

   例:

   ```
   export DB2DBDFT=CWPortal
   ```

7. pin をログオフする。

ユーザー「cwadmin」として

1. コネクターの始動に使用されるユーザー cwadmin として、AIX システムにログインする。
2. シェル・スクリプト・ファイル db2profile を実行してから、コネクターを始動する。

   このファイルは、DB2 用 Portal Infranet のインストールに使用したアカウントのホーム・ディレクトリー内のディレクトリー sqllib、例えば、/home/db2instl/sqllib に存在します。

   ファイル connector_manager_Portal を $*ProductDir*/connectors/Portal から $*ProductDir*/connectors/bin にコピーする。

3. ディレクトリーを $*ProductDir*/connectors/Portal に変更する。ファイル start_Portal.sh で、以下の行を

   PORTAL_HOME=/opt/portal/6.1

   以下のように、適切な値に変更する。

   PORTAL_HOME=PortalInfranet_home_dir

   例:

   /opt/portal/6.2
4. cwadmin をログオフする。

ユーザー「pin」として

1. pin として、AIX システムにログインする。
2. Infranet ファイルの pcm.jar および pcmext.jar を $*ProductDir*/connectors/Portal/dependencies にコピーする。この 2 つのファイルは、Infranet サーバーの $INFRANET/jars ディレクトリーに存在します。
3. ファイルの custom_opcode.h を $INFRANET/include にコピーする。
4. ファイルの custom_opcode_mem_map を $INFRANET/lib にコピーする。
5. .profile ファイルを pin のホーム・ディレクトリー、例えば、/home/pin にコピーする。必要な場合は、.profile ファイルを変更して、使用するシステムに設定された環境変数を反映します。どのような変更の場合も、vi などのテキスト・エディターを使用してください。環境変数が正しい場合は、コマンド・プロンプトから、以下のコマンドを入力して、環境変数をシステムにロードします。

   ../.profile
6. ファイルの fm_crossworlds.so を $INFRANET/lib ディレクトリーにコピーする。このファイルにはイベント用のトリガーが含まれます。

   注: AIX は大文字小文字を区別するため、ファイルが見つからない場合には、すべてのディレクトリー名およびファイル名に大文字小文字が正しく指定されているかを確認してください。

   $LIBRARY_PATH 変数に $INFRANET/lib パスが含まれていてシステムが .so ファイルを認識できるかどうかを確認してください。
7. $LIBRARY_PATH 変数に $INFRANET/lib パスが含まれていてシステムが .so ファイルを認識できるかどうかを確認してください。
8. 以下のファイルをディレクトリー $CW_PORTAL_PATH、例えば、/opt/portal/6.2/sys/cm にコピーする。
   - crossworlds.cnf: このファイルには、イベント・モジュールの構成情報が含まれています。
   - event_code.txt: このファイルには、イベント・モジュールが WebSphere Business Integration システムのイベント表内にエントリーを生成するために使用する Infranet イベントの記述が含まれています。
9. pin_notify_cw ファイルを $INFRANET/sys/test ディレクトリーに配置します。このファイルにはコネクター・イベントの名前が含まれます。追加または削除すべきイベントがある場合は、このファイルの標準形式に従ってください。/event は /event/customer や /event/status などのすべてのサブクラスをカプセル化することに注意してください。

AIX/DB2 環境での Portal Infranet アプリケーションの構成

DB2 をデータベース、AIX をオペレーティング・システムとして使用している場合は、以下の手順を実行して、Portal Infranet アプリケーションを構成します。

1. Infranet アプリケーションを停止して再始動します。$INFRANET/bin が $PATH 変数内にあるかどうかを確認してください。以下のステップを実行してください。
   a. 次のコマンドで Infranet を停止します。

      ```
      stop_all
      ```

   b. 次のコマンドを入力して、すべての Infranet プロセスが停止したかどうか確認します。実行中の Infranet プロセスがある場合は、そのプロセス番号 (PID) を控えます。

      ```
      ps -ef | grep protal
      ```

   c. 次のコマンドを入力して、実行中の Infranet プロセスをすべて強制終了します。

      ```
      kill -9 PID
      ```

   d. 次のコマンドで Infranet を再始動します。

      ```
      start_all
      ```

2. $CM ディレクトリーで、pin.conf ファイルの fm_required セクションに以下の行を追加します。

   ```
   -cm fm_module $INFRANET/lib/fm_crossworlds.so fm_cw_config -pin
   ```

   必ず、$INFRANET の完全なディレクトリー・パスを入力してください。
3. 以下のコマンドを入力して、Infranet が実行中であるかどうか検証します。

   ```
   ps -ef | grep portal
   ```

4. $INFRANET/sys/test にディレクトリーを変更して、pin.conf ファイルを開き、以下のような行が含まれているかどうか確認します。

   ```
   nap_cm_ptr ip Infranet_cm_machine cm_port
   ```

   例:

   ```
   nap_cm_ptr ip roadrunner 11960
   ```

   ここで、roadruner は *Infranet_cm_machine*、11960 は *cm_port* です。
5. pin.conf ファイルに以下の内容が含まれていることを確認します。
6. `nap login_type 1 nap login_name root.0.0.0.1 nap login_pw password`

   これは、Infranet に接続するためのログイン情報を示します。ディレクトリーに pin.conf ファイルが存在しない場合は、ディレクトリー内に同ファイルをコピーしてください。
7. 以下のコマンドを入力して、Infranet アプリケーションに構成情報をロードします。

   ```
   load_pin_notify pin_notify_cw
   ```

   応答は successful でなければなりません。他の応答が表示された場合は、pin_notify_cw をチェックしてください。このファイルには、特定のイベント発

生時に Infranet が呼び出す命令コードが含まれています。ここで、pin_notify_cw は load_pin_notify 実行可能ファイルと同じディレクトリー内に配置してください。

8. $INFRANET_VAR/cm ディレクトリーで、ログ・ファイルをチェックして、$CM にコア、例えば、cm.pinlog に cm started at "Tue Jul 9 20:49:37 2002" が存在することを確認します。Infranet Administrator を始動します。
9. 1 つのアカウントを入力または変更し、イベント表 xworlds_events に適切なイベント・エントリーが含まれるかどうかを確認することにより、コネクターをテストします。このテストの結果はダミー・イベントとなるため、テストが完了したらイベント・エントリーを削除します。

Oracle 環境でのアダプターの構成

Oracle をデータベースとして使用している場合は、以下の手順を実行して、コネクターを構成します。

1. pin として UNIX システムにログインします。必要な場合は、このアカウントを作成します。
2. Infranet ファイル pcm.jar および pcmext.jar を $*ProductDir*/Connector/Portal/dependencies にコピーします。このファイルは、Infranet 6.1.0 サーバーの nfranet/jars ディレクトリーに配置されます。
3. .profile ファイルを pin ユーザーのホーム・ディレクトリー、例えば、/home/pin にコピーします。必要な場合は、.profile ファイルを変更して、使用するシステムに設定された環境変数を反映します。どのような変更の場合も、vi などのテキスト・エディターを使用してください。

   環境変数が正しい場合は、コマンド・プロンプトから以下のコマンドを入力して環境変数をシステムにロードします。

   ```
   source .profile
   ```

4. fm_crossworlds.so ファイルを $INFRANET/lib ディレクトリーに置きます。このファイルにはイベント用のトリガーが含まれます。

   UNIX は大文字小文字を区別するため、ファイルが見つからない場合には、すべてのディレクトリー名およびファイル名に大文字小文字が正しく指定されているかを確認してください。
5. $LIBRARY_PATH 変数に $INFRANET/lib パスが含まれていてシステムがコネクター .so ファイルを認識できるかどうかを確認してください。
6. 以下のファイルを $CW_PORTAL_PATH ディレクトリーにコピーします。
   - crossworlds.cnf: このファイルには、イベント・モジュールの構成情報が含まれます。
   - event_code.txt: このファイルには、イベント・モジュールが WebSphere Business Integration Adapter イベント表内にエントリーを生成するために使用する Infranet イベントの記述が含まれます。
7. pin_notify_cw を $INFRANET/sys/test ディレクトリーに配置します。このファイルにはコネクター・イベントの名前が含まれます。追加または削除すべきイベントがある場合は、このファイルの標準形式に従ってください。/event は /event/customer や /event/status などのすべてのサブクラスをカプセル化することに注意してください。

8. Infranet アプリケーションを停止して再始動します。$INFRANET/bin が $PATH 変数内にあるかどうかを確認してください。以下のステップを実行してください。
   a. 次のコマンドで Infranet を停止します。

      ```
      stop_all
      ```

   b. 次のコマンドを入力して、すべての Infranet プロセスが停止したかどうか確認します。実行中の Infranet プロセスがある場合は、そのプロセス番号 (PID) を控えます。

      ```
      ps -ef|grep portal
      ```

   c. 次のコマンドを入力して、実行中の Infranet プロセスをすべて強制終了します。

      ```
      kill -9 <PID>
      ```

   d. 次のコマンドで Infranet を再始動します。

      ```
      start_all
      ```

9. $CM ディレクトリーで、pin.conf ファイルを編集して以下の行を fm_required セクションに追加します。必ず、$INFRANET の完全なディレクトリー・パスを入力してください。

   ```
   - cm fm_module $INFRANET/lib/fm_crossworlds.so fm_cw_config -pin
   ```

10. コマンド ps -ef|grep portal を入力して、Infranet が実行中であるかどうか検証します。
11. $INFRANET/sys/test ディレクトリーに変更して pin.conf ファイルを開き、以下のような行が含まれるかどうか確認します。

    ```
    - nap cm ptr ip Infranet_cm_machine cm_port
    ```

    例えば、次のような行です。

    ```
    - nap cm ptr ip roadrunner 11960
    ```

    ここで、roadrunner は *Infranet_cm_machine*、*cm_port* は 11960 です。

    上記のステートメント以外にも、pin.conf ファイルには以下の行が含まれる必要があります。

    ```
    - nap login_type 1
    - nap login_name root.0.0.0.1
    - nap login_pw password
    ```

    これらは、Infranet に接続するためのログイン情報を示します。ディレクトリーに pin.conf ファイルが存在しない場合、ディレクトリー内に同ファイルをコピーしてください。

12. Infranet アプリケーションに構成情報をロードするため、以下のコマンドを入力します。

    ```
    load_pin_notify pin_notify_cw
    ```

    応答は successful でなければなりません。他の応答が返された場合は、pin_notify_cw を確認してください。このファイルには、特定のイベント発生時に Infranet が呼び出す命令コードが含まれます。ここで、pin_notify_cw は load_pin_notify 実行可能ファイルと同じディレクトリー内に配置してください。

13. $INFRANET_VAR/cm ディレクトリーで、ログ・ファイルをチェックして $CM にコアが含まれることを確認し、Infranet Administrator を始動します。
14. 1 つのアカウントを入力または変更し、イベント表 xworlds_events に適切なイベント・エントリーが含まれるかどうかを確認することにより、コネクターをテストします。このテストの結果はダミー・イベントとなるため、テストが完了したらイベント・エントリーを削除する必要があります。

コネクターを始動する場合は、 32 ページの『コネクターの始動』を参照してください。

Windows 環境に対するイベント機構のインストールと構成

Infranet には、アプリケーション内で発生するアクションの追跡を可能にするイベント機構が備わっています。コネクターがイベント・デリバリーを処理できるようにするには、あらかじめ Infranet にイベント通知機構を構成しておく必要があります。以下のステップを実行してください。

1. 以下の WebSphere Business Integration Adapter イベント・モジュール・ファイルをコピーします。
   - FmCw.dll を %INFRANET%¥lib にコピーします。この DLL はコネクターのイベント機能モジュールです。
   - event_code.txt を %INFRANET%¥sys¥cm にコピーします。このファイルは Infranet イベントをイベント・モジュールに記述し、イベント・モジュールは Infranet イベントに対応するエントリーを WebSphere Business Integration Adapter イベント表に生成します。

     ここで、%INFRANET% は Infranet ホーム・ディレクトリー内にあることに注意してください。
   - custom_opcode.h を %INFRANET%¥include にコピーします。
   - custom_opcode_mem_map を %INFRANET%¥lib にコピーします。
   - Infranet ファイルの pcm.jar および pcmext.jar を $CROSSWORKDS/Connector/Portal/dependencies にコピーします。このファイルは、Infranet サーバーの infranet/jars ディレクトリーに配置されます。
2. %INFRANET%¥sys¥cm で、接続マネージャーの pin.conf 構成ファイルを編集して接続マネージャーがコネクター・イベント機能 DLL FmCw.dll を実行できるようにします。pin.conf に以下の行を追加します。

   ```
   - cm fm_module ..¥..¥lib¥FmCw.dll fm_cw_pol_config_func - pin --
   ops_fields_extension_file ..¥..¥lib¥custom_opcode_mem_map
   ```
3. pin_notify_cw ファイルを %INFRANET%¥sys¥test ディレクトリーに配置します。このファイルは、特定のイベント発生時に Infranet が呼び出す命令コードを指定します。追加または削除すべき命令コードがある場合は、このファイルの標準形式に従ってください。/event は /event/customer や /event/status などのすべてのサブクラスをカプセル化することに注意してください。
4. %INFRANET%¥sys¥test 内の pin.conf ファイルを以下のように更新します。

   ```
   - nap cm ptr ip <Infranet_cm_machine> <port>
   ```

   例えば、次のような行です。

   ```
   - nap cm ptr ip cwengtest 11960
   ```

ここで、cwengtest は <Infranet_cm_machine>、<port> は 11960 であることに注意してください。

上記のステートメント以外にも、pin.conf ファイルには以下の行が含まれる必要があります。

```
- nap login_type 1
- nap login_name root.0.0.0.1
- nap login_pw password
```

これらは、Infranet に接続するためのログイン情報を示します。ディレクトリーに pin.conf ファイルが存在しない場合、ディレクトリー内に同ファイルをコピーしてください。

5. crossworlds.cnf 環境構成ファイルを %INFRANET%¥sys¥cm にコピーします。このファイルは、イベント機能モジュール FmCw.dll が環境についての情報を入手できるようにします。

   必要な場合は、このファイルを使用するシステムに応じて編集してください。crossworlds.cnf の内容の例を以下に示します。

   ```
   db user = pin
   db password = pin
   crossworlds id = 0.0.0.1 ¥service¥admin_client 14088
   log level = 3
   log file = D:¥pinlog.log
   ```

   ここで、以下のように説明されます。

db user	Portal Infranet データベースに接続しているユーザーの名前。
db password	パスワード。
crossworlds id	Portal Infranet 内の WebSphere Business Integration Adapter ユーザーを表す POID。
log level	次のようにログ・レベルを表す数字。 0 : トレースなし 1 : エラーのみ 2 : エラーと警告 3 : エラー、警告、およびデバッグ (すべてトレースする)
log file	ログ・ファイルの名前。

   このファイルはハッシュ・テーブルと見なされます。キーワードは等号の左のストリングで、値は右のストリングです。

6. crossworlds.cnf ファイルへのパスを指定する CW_PORTAL_PATH という環境変数を作成します。
7. Infranet サービスを以下の順序で再始動します。
   - Infranet データ・マネージャー
   - Infranet Java サーバー
   - Infranet 接続マネージャー
8. 以下の構文で pin_notify_cw ファイルを Infranet アプリケーションにロードします。

```
load_pin_notify .¥pin_notify_cw
```

エラーが発生する場合は、¥bin ディレクトリーが PATH に含まれて、実行可能ファイルを配置できるようになっているかどうか確認してください。エラー・メッセージは、実行ディレクトリーの pinlog.log で確認してください。

コネクターの構成

アダプターの構成プロパティーには、標準構成プロパティーとアダプター固有の構成プロパティーという 2 つのタイプがあります。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。

コネクター・プロパティーは Connector Configurator から設定します (WebSphere MQ Integrator Broker が統合ブローカーである場合)。あるいは、コネクター・プロパティーを Connector Configurator から設定し、Connector Configurator には System Manager からアクセスします (ICS が統合ブローカーである場合)。構成情報の詳細については、95 ページの『付録 B. Connector Configurator』または「*コネクター開発ガイド (Java 用)*」を参照してください。

標準コネクター・プロパティー

標準構成プロパティーは、すべてのコネクターが使用する情報を提供します。これらのプロパティーの資料については、77 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。

重要: このコネクターはすべての統合ブローカーをサポートするため、すべてのブローカーの構成プロパティーはそのコネクターに関係があります。

注: このコネクターは単一スレッドであるため、AgentConnections プロパティーを利用することができません。

表 2 には、付録の標準構成プロパティーの中で、このコネクターに特定の情報に示します。

表 2. このコネクターに特定の標準プロパティー情報

プロパティー	注
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可能です。現在サポートされているロケールを確認するには、コネクターのリリース情報を参照してください。

重要: WebSphere MQ Integrator Broker は、複数のロケールをサポートしません。インストールされているすべてのコンポーネント (例えば、すべてのアダプター、アプリケーション、および統合ブローカー自体) に、同一のロケールがセットされていることを確認します。

注: このコネクターはすべての統合ブローカーをサポートするため、すべてのブローカーの構成プロパティーはそのコネクターに関係があります。

コネクター固有のプロパティー

コネクター固有の構成プロパティーは、コネクターが実行時に必要とする情報を提供します。また、コネクター固有の構成プロパティーを使用すると、コネクターのコード変更や再構築を行わなくても、コネクター内の静的情報またはロジックを変更できます。

表 3 に、コネクターのコネクター固有の構成プロパティーのリストを示します。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 3. コネクター固有の構成プロパティー

名前	可能な値	デフォルト値	必須
Application password	*ユーザー・アカウントのパスワード*		はい
ApplicationUserName	*ユーザー・アカウントの名前*		はい
CommentFail	*失敗したイベントに関するイベント表のコメント*	Fail	いいえ
CommentSucceed	*正常なイベントに関するイベント表のコメント*	Succeed	いいえ
DatabaseInfo	jdbc:oracle:thin@CWENGTEST:1521:portaldb	サーバー・ホスト名、Portal データベース・ポート番号、およびデータベースの名前	はい
dbDriverClass (DB2 データベースのみ)	*データベースのドライバー・クラス名*		はい
DbName	*Oracle Listener 名*		はい
DbPassword	*データベース・パスワード*		はい
dbURL (DB2 データベースのみ)	*データベースの URL*		はい
DbUser	*データベース・ユーザー名*	pin	はい
InfDatabase	*ストリング*	0.0.0.1	はい
InfHost	*ホストのマシン名およびポート*	//CWENGTEST1:11960	はい
InfLogFile	*ログ・ファイルの名前*	InfConnection.txt	はい
InfranetConnections	*コネクターが接続プールのために開いた Infranet 接続の数*	5	はい
InfService	*ストリング*	service¥admin_client 1	はい
InfType	0 または 1	1	はい

表 3. コネクター固有の構成プロパティー (続き)

名前	可能な値	デフォルト値	必須
InfVersion	*Infranet* アプリケーションのバージョン		はい
PollQuantity	*選出*するイベントの数	1	いいえ
UseDefaults	`true` または `false`	`false`	いいえ

Application password
WebSphere Business Integration システムのユーザー・アカウントのパスワードです。

ApplicationUserName
コネクターのユーザー・アカウント名。

CommentFail
イベントが失敗した場合のイベント表のコメントです。デフォルト値は `Fail` です。

CommentSucceed
イベントが成功した場合のイベント表のコメントです。デフォルト値は `Succeed` です。

DatabaseInfo
このプロパティーは、JDBC ドライバーがデータベースに接続するために使用する URL を定義します。

例:

```
jdbc:oracle:thin:@CWENGTEST1:1521:portaldb
```

URL の特定のフォーマットについては、JDBC の資料を参照してください。

dbDriverClass
データベースのドライバー・クラス名です (DB2 のみ)。`dbDriverClass` および `dbURL` が関連値を持つ場合、コネクターは、それらの値を使用して、`dbName` および `DatabaseInfo` を無視します。この 2 つのプロパティーの一方に、`null` がセットされている場合、`dbName` および `DatabaseInfo` が読み取られ、コネクターはデータベースが Oracle であるものと仮定します。DB2 には、`COM.ibm.db2.jdbc.app.DB2Driver` をセットします。

DbName
Infranet データベース用の Oracle Listener 名です。

DbPassword
データベース・パスワードです。

dbURL
データベースの URL です (DB2 のみ)。dbDriverClass および dbURL が関連値を持つ場合、コネクターは、それらの値を使用して、dbName および DatabaseInfo を無視します。この 2 つのプロパティーの一方に、null がセットされている場合、dbName および DatabaseInfo が読み取られ、コネクターはデータベースが Oracle であるものと仮定します。jdbc:db2:*DB_name*、例えば、jdbc:db2:CWPortal をセットします。

DbUser
データベース・ユーザー名、通常は pin。

InfDatabase
Infranet 形式のストリングです。デフォルト値は、0.0.0.1. です。

InfHost
ホスト・マシン名とポート、例えば、 //engtest2:11960。デフォルト値は、//CWENGTEST1:11960 です。

InfLogFile
デフォルト・ログ・ファイルとして使用するファイルの名前です。デフォルト値は、InfConnection.txt です。

InfranetConnections
このプロパティーは、コネクターが、接続プール用に Infranet アプリケーションで開く接続の数を定義するために使用します。コネクターは、割り当てられた接続の数を維持します。接続を必要とするビジネス・オブジェクト・プロセスは、接続プールから 1 つの接続を割り当てます。その接続は、使用可能プールから除去され、使用中プールに追加されます。ビジネス・オブジェクト・プロセスが完了すると、接続は使用中プールから除去され、使用可能プールに戻されます。このようにして、接続プールを使用すると、各ビジネス・プロセスに対して、接続のオープン・クローズを行う必要がなくなるため、性能効率が向上します。

注: このプロパティーがセットされていない場合、コネクターは、始動されたときに、例外の NumberFormatException をスローします。

InfService
ストリング、通常は service¥admin_client 1。これはデフォルト値です。

InfType
接続タイプです。可能な値は 0 および 1 のみです。デフォルト値は 1 です。

InfVersion
Infranet アプリケーションのバージョンです。

PollQuantity
1 回のポーリングで選出されるイベントの数です。デフォルト値は 1 です。

UseDefaults
UseDefaults を true に設定したかまたは省略した場合に、isRequired ビジネス・オブジェクト属性に対して有効な値またはデフォルト値が指定されているかどうかを

コネクターがチェックします。値が指定されている場合は Create が正常に終了します。指定されていない場合は失敗します。

このパラメーターを false に設定した場合は、コネクターが有効な値のみをチェックします。有効な値が指定されていない場合は Create 操作が失敗します。

デフォルト値は false です。

新規ビジネス・オブジェクト用のイベント機構のカスタマイズ

新規ビジネス・オブジェクトを作成する場合は、そのビジネス・オブジェクトのそれぞれのアクションごとに生成されるイベントを決定する必要があります。イベントを決定したら、イベント・モジュール構成ファイルをカスタマイズして、イベント・モジュール DLL がそのタイプのイベントを検出できるようにする必要があります。イベント・モジュール構成ファイルの名前は `event_code.txt` で、このファイルは `$INFRANET$¥sys¥cm` 内にあります。

Infranet は、あるイベントで格納可能クラスのどの部分が (すなわちどのビジネス・オブジェクトが) 起動されたのかをイベント・モジュールが確認するために必要なデータを生成します。イベント・モジュールは、イベントを取得する際に、アカウント、ユーザー、および呼び出し側プログラムなどの情報を含む格納可能クラスのインスタンスを取得します。

更新を削除や作成と区別するために、イベント・モジュールは元の値と更新された値を比較します。しかし、作成または削除が発生したかどうかを検出するためには、イベント・モジュールはルート・レベルでアクションが実行された時に格納可能クラスを検索するか、エレメント ID を調べて子オブジェクトの有無を判断する必要があります。子オブジェクトが追加されている場合、エレメント ID は正であり、配列内でのそのエレメントの位置を含みます。エレメント ID が負の場合、そのエレメントは除去されています。

イベント・モジュール構成ファイルの構文

イベント・モジュール構成ファイルを変更する場合には、ファイルに対する変更が以下の構文規則に準拠していなければなりません。この構文は厳守する必要があります。

1. コメント行は 2 つのダッシュで開始すること。
2. 1 つのイベントは 1 行に記述すること。パラメーターはパイプ文字 (|) で区切ること。
3. この構文に準拠すると、次のようになります。

```
<event>|<Inf.action>|<array>|<key poid>|<constraints>|<BO.verb>|<priority>|<comment>
```

ここで、以下のように説明されます。

event	イベントの格納可能クラス名。
Infranet action	C (作成)、U (更新)、または D (削除)。Portal Infranet 内で実行されるアクションを表します。
array	アクションが実行される配列を表す Portal Infranet コード。配列は、イベント情報から検索する必要がある Infranet エレメントです。 Portal Infranet では、各フィールドには 1 つの番号が関連付けられています。例えば、PIN_FLD_NAMEINFO は Infranet コード 156 に関連付けられています。フィールドおよび関連付けられた番号のリストは、ファイル $INFRANET$¥Include¥pin_flds.h を参照してください。
poid	作成、更新、または削除された格納可能クラスのキーとなるフィールドを表す Portal Infranet コード。
constraints	何が起こったかを正確に判断するために必要な制約のリスト。制約は以下のキーワードをサポートします。 • exists または not_exists: オブジェクトが存在するかしないかを指定する真偽。 • =、>、>=、<=、< : 数値型の比較演算子 • equal、nequal、contains: ストリング型のための比較演算子。 • & は、制約を区切るために使用します。& で区切られたすべての条件が真である場合のみ、その制約が真になります。 or 条件を指定する場合は、複数行を使用してください。制約が真である最初の行が実行されます。
BO.verb	Portal Infranet 内のアクションに対応するビジネス・オブジェクトおよび動詞の名前。
priority	イベントの優先順位。
comment	イベント表に挿入するコメント。

イベント・モジュール構成ファイルの例

以下のテキストは、event_code.txt ファイルの一例です。この例には、account 格納可能クラスのイベントを指定する行が含まれます。ダッシュで始まる行はコメントです。

```
-- Account creation: PIN_FLD_STATUSES[0].PIN_FLD_STATUS[0] = 0 &
      PIN_FLD_SYS_DESCR = "Set Status (acct)": OK
/event/customer/status |U |144  |40  |40 exists&55;5 equal "Set Status
      (acct)"&144:0-145;3 = 0 |Portal_Account.Create |1 |Account Creation

-- Account Updated (status updated) : PIN_FLD_STATUSES[1].PIN_FLD_STATUS[0] =
      10100 or 10103 or 10102 & & PIN_FLD_SYS_DESCR = "Set Status (acct)" : OK
/event/customer/status |U |144  |40  |40 exists&55;5 equal "Set Status (acct)
      "&144:1-145;3 > 0 |Portal_Account.Update |1 |Status Updated

-- Account Updated (new contact added):OK
/event/customer/nameinfo |C |156 |40|40 exists|Portal_Account.Update|1|new contact

-- Account Updated (contact updated): OK
/event/customer/nameinfo |U |156 |40 |40 exists&17;5 equal "Customer Mngmt. Event
      Log" |Portal_Account.Update |2 |contact_update

-- Account Updated (contact deleted):OK
/event/customer/nameinfo |D |156  |40  |40 exists&67;5 nequal "Automatic Account
      Creation"|Portal_Account.Update |2 |contact_delete
```

```
-- Account Updated (billinfo updated) : two PIN_FLD_BILLINFO and
      PIN_FLD_BILLINFO[0].PIN_FLD_BILL_TYPE[0] <> 0 : OK
/event/customer/billinfo |U |126  |40 |40 exists & 126:0-127;3 > 0
      Portal_Account.Update |2 |billinfo_update
```

イベント構成ファイルのエントリーの定義

あるイベントが発生すると、Infranet はそのイベントを表現する flist (フィールド・リスト) を生成します。イベント機能モジュールはフィールド・リストを検査して、動詞および発生したアクションを識別します。

例えば、あるイベントを表すフィールド・リストを想定すると以下のようになります。

```
0     PIN_FLD_POID            POID [0] 0.0.0.1 /event/customer/nameinfo -1 0
0     PIN_FLD_NAME             STR [0] "Customer Mngmt. Event Log"
0     PIN_FLD_USERID          POID [0] 0.0.0.1 /service/admin_client 2 1
0     PIN_FLD_SESSION_OBJ     POID [0] 0.0.0.1 /event/session 10366 0
0     PIN_FLD_ACCOUNT_OBJ     POID [0] 0.0.0.1 /account 9406 32
0     PIN_FLD_PROGRAM_NAME     STR [0] "Admin Manager"
0     PIN_FLD_START_T       TSTAMP [0] (942104217) 11/08/99 15:36:57
0     PIN_FLD_END_T         TSTAMP [0] (942104217) 11/08/99 15:36:57
0     PIN_FLD_SYS_DESCR        STR [0] "Set Name Info"
0     PIN_FLD_NAMEINFO       ARRAY [0] allocated 20, used 17
1     PIN_FLD_SALUTATION       STR [0] ""
1     PIN_FLD_LAST_NAME        STR [0] "Event Test1"
1     PIN_FLD_LAST_CANON       STR [0] "event test1"
1     PIN_FLD_FIRST_NAME       STR [0] "Event Test1"
1     PIN_FLD_FIRST_CANON      STR [0] "event test1"
1     PIN_FLD_MIDDLE_NAME      STR [0] ""
1     PIN_FLD_MIDDLE_CANON     STR [0] ""
1     PIN_FLD_TITLE            STR [0] "Event Test1  "
1     PIN_FLD_COMPANY          STR [0] "Event Test1"
1     PIN_FLD_ADDRESS          STR [0] "Event Test1"
1     PIN_FLD_CITY             STR [0] "Event Test1"
1     PIN_FLD_STATE            STR [0] "CA"
1     PIN_FLD_ZIP              STR [0] "00000"
1     PIN_FLD_COUNTRY          STR [0] "US"
1     PIN_FLD_EMAIL_ADDR       STR [0] ""
1     PIN_FLD_CONTACT_TYPE     STR [0] "Billing"
1     PIN_FLD_ELEMENT_ID      UINT [0] 1
0     PIN_FLD_NAMEINFO       ARRAY [1] allocated 20, used 19
1     PIN_FLD_SALUTATION       STR [0] ""
1     PIN_FLD_LAST_NAME        STR [0] "Event Test1"
1     PIN_FLD_LAST_CANON       STR [0] "event test1"
1     PIN_FLD_FIRST_NAME       STR [0] "Event Test1"
1     PIN_FLD_FIRST_CANON      STR [0] "event test1"
1     PIN_FLD_MIDDLE_NAME      STR [0] ""
1     PIN_FLD_MIDDLE_CANON     STR [0] NULL str ptr
1     PIN_FLD_TITLE            STR [0] "Event Test1"
1     PIN_FLD_COMPANY          STR [0] "Event Test1"
1     PIN_FLD_ADDRESS          STR [0] "Event Test1"
1     PIN_FLD_CITY             STR [0] "Event Test1"
1     PIN_FLD_STATE            STR [0] "CA"
1     PIN_FLD_ZIP              STR [0] "00000"
1     PIN_FLD_COUNTRY          STR [0] "US"
1     PIN_FLD_EMAIL_ADDR       STR [0] ""
1     PIN_FLD_CONTACT_TYPE     STR [0] "Billing"
1     PIN_FLD_ELEMENT_ID      UINT [0] 1
1     PIN_FLD_CANON_COUNTRY    STR [0] "US"
1     PIN_FLD_CANON_COMPANY    STR [0] "event test1"
0     PIN_FLD_ITEM_OBJ        POID [0] 0.0.0.1 /item 11454 0
0     PIN_FLD_CURRENCY        UINT [0] 840
```

イベント・モジュール・ファイルに行を追加する場合は、フィールド名ではなくフィールドに関連付けられた番号を指定する必要があります。例えば、前項のフィールド・リストで太字で示されているフィールド PIN_FLD_NAMEINFO は、コード 156 によって表されます。

フィールド・リストの第 1 レベルよりも深いフィールドの場合は、制約を使用してそのフィールドを識別する必要があります。制約の例外として、各フィールドは、そのフィールドに関連付けられた単一番号で識別されます。

特定のイベントはさまざまな理由で生成されるため、あるイベントが特定のビジネス・オブジェクトの結果かどうかを判断するために制約が必要になる場合があります。したがって、exists などの制約を追加しなければならない可能性があります。

制約の中では、以下の構文を使用してフィールドを識別する必要があります。

```
[<array code:array element>-field;type]
```

例えば、Infranet コード 156 が PIN_FLD_NAMEINFO を表し、Infranet コード 161 が PIN_FLD_LAST_NAME を表す場合、エレメント ID 1 の配列 PIN_FLD_NAMEINFO に含まれるフィールド PIN_FLD_LAST_NAME の表記は、このフィールドに関連する制約で表すと 156:1-161;5 = PINFLDNAMEINFO[1].PIN_FLD_LASTNAME となります。

制約内のセミコロンの後の最後の番号は、オブジェクトのタイプを表します。このタイプは、比較を実行する必要があるデータのタイプを示しています。例えば、番号 5 はストリングを表します。詳細については、$INFRANET$¥Include¥pin_type.h に含まれる関連タイプのリストを参照してください。

pin_notify_cw ファイルへのイベントの追加

新規ビジネス・オブジェクトを追加する場合には、pin_notify_cw ファイルにもイベントを追加する必要があります。このファイルにはコネクター・イベントの名前が含まれます。イベントを追加する際には、ファイルの標準形式に従ってください (28 ページの『イベント構成ファイルのエントリーの定義』を参照)。

CrossWorlds イベント通知モジュール (FmCw.dll) は、自身に送られてきたすべてのポータル・イベントのユーザー ID を確認します (pin_notify_cw ファイルで定義)。モジュールに送られてきたイベントに対応する PIN_FLD_USERID がない場合はエラーとなり、オブジェクトのオンライン保存時に問題が発生します。このようなタイプのイベントは、正しい ID を組み込むための調整を行う必要があります (フィールド・リストや格納可能クラスを使用)。crossworlds.cnf 構成ファイルで定義されたログ・ファイルにこれらのエラーがないか確認してください。

イベント・モジュールは、ユーザー ID を調べることにより、コネクターによってアプリケーションに送信されたイベントがイベント・キューに追加されるのを防ぎます。これを「ピンポン」と呼びます。/event/customer/billinfo は、この種の問題のあるイベント・タイプの例です。

注: ある Infranet 操作のために生成されたすべてのイベントを確認するには、すべての Infranet イベントのイベント・モジュールを呼び出してください (load_pin_notify 構成ファイルの /event を使用)。すると、発生しているすべてのイベントに対してイベント・モジュールが起動されます。イベント・モジ

ュール構成ファイルのログ・レベルを 3 に設定してから pinlog.log ファイルを開くと、Infranet により送信されたこの操作に関するイベント・フィールド・リストが生成されています。

Infranet カスタム属性のオプショナル構成の宣言

Infranet アプリケーションのカスタマイズが完了したら、対応する Java クラスを生成する必要があります。Infranet はこれらのクラスを生成するためのツールを提供します。コネクターは実行時にこれらのクラスを認識する必要があるため、CLASSPATH でクラスを宣言する必要があります。

1. あるディレクトリーを作成します。(例: c:¥CustomAttributes)
2. カスタム・フィールドとその値を持つ #include ファイルを作成します。
3. このファイルで custom_fields.pl スクリプトを起動して、com.portal.pcm.fields をパッケージ名として表示します。
4. 以下のように Java ファイルをコンパイルして、クラス・ファイルを作成します。

   ```
   javac -classpath c:¥program files¥infranet¥java¥pcmext.jar *.java
   ```

5. 現行ディレクトリーの下にディレクトリー階層 com¥portal¥pcm¥fields を作成します。
6. コンパイラーで作成された .class をこのディレクトリーにコピーします。
7. 現行ディレクトリー (例えば、c:¥CustomAttributes) をコネクターの CLASSPATH に追加します。
8. 必要であれば、¥bin¥start_Portal.bat ファイルを変更します。

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。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 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 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、 30 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 (Windows のみ)。
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。

カスタム・コネクター作成の詳細については、「*コネクター開発ガイド (C++ 用)*」または「*コネクター開発ガイド (Java 用)*」を参照してください。

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 4 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 4. 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
UNIX ベースのシステム	connector_manager_*connName*
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 Adapters」>「アダプター」>「コネクター」 を選択します。デフォルトでは、プログラム名は「IBM WebSphere Business Integration Adapters」となっています。ただし、これはカスタマイズすることができます。あるいは、ご使用のコネクタ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 brokerName* [-c*configFile*]
  - UNIX ベースのシステム:

    connector_manager_*connName* -start

  ここで、*connName* はコネクターの名前であり、*brokerName* は以下のようにご使用の統合ブローカーを表します。
  - WebSphere InterChange Server の場合は、*brokerName* に ICS インスタンスの名前を指定します。
  - WebSphere Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、または WebSphere Business Integration Message Broker) または WebSphere Application Server の場合は、*brokerName* にブローカーを示すストリングを指定します。

  注: Windows システム上の WebSphere Message Broker または WebSphere Application Server の場合は、-c オプションに続いてコネクター構成ファイルの名前を指定しなければなりません。ICS の場合は、-c はオプションです。
- Adapter Monitor から (WebSphere Business Integration Adapters 製品のみ)。Adapter Monitor は System Manager 始動時に起動されます。

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

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

  このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。
- Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムがブートしたとき (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、以下の資料のいずれかを参照してください。

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

コネクターの停止

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。
  - UNIX ベースのシステムでは、コネクターはバックグラウンドで実行されるため、別ウィンドウはありません。代わりに、次のコマンドを実行してコネクターを停止します。

    ```
    connector_manager_connName -stop
    ```

    ここで、*connName* はコネクターの名前です。
- Adapter Monitor から (WebSphere Business Integration Adapters 製品のみ)。Adapter Monitor は System Manager 始動時に起動されます。

  このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。
- System Monitor から (WebSphere InterChange Server 製品のみ)

  このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。
- Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムのシャットダウン時に、コネクターは停止します。

第 3 章 ビジネス・オブジェクトの理解

この章では、コネクターがビジネス・オブジェクトを処理する方法を説明し、Portal 用ビジネス・オブジェクトを実装するためのガイドラインを提供します。以下の項目について説明します。

- 『Portal Infranet アプリケーションのバックグラウンド』
- 39 ページの『Portal Infranet アプリケーション固有のビジネス・オブジェクト構造』
- 43 ページの『ビジネス・オブジェクトの定義に関するガイドライン』
- 43 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』

注: Infranet アプリケーションについて十分な知識と経験がない場合は、Portal Infranet 用ビジネス・オブジェクトを設計および実装することは困難です。この場合は、アプリケーションのエキスパートとともに作業することをお勧めします。Infranet の概念またはプログラミング・エレメントについては、Infranet 資料を参照してください。

Portal Infranet アプリケーションのバックグラウンド

このセクションでは、Portal Infranet アプリケーション固有のビジネス・オブジェクトの設計およびインプリメンテーションに影響する、Infranet アプリケーションのいくつかの基本エレメントについて、簡単な概要を示します。Infranet は、システム内の機能を定義または拡張したり、機能にアクセスしたりするために使用される 4 つの主要なプログラミング・エレメントを定義します。Portal 用ビジネス・オブジェクトを設計するには、次のエレメントについてよく理解している必要があります。以下のセクションでは、これらについて簡単に説明します。

- 格納可能クラス
- 格納可能オブジェクト
- フィールドおよびフィールド・リスト (flist)
- 命令コード

格納可能クラスと格納可能オブジェクト

Infranet では、格納可能クラスにはクラスについての情報を保管するフィールドが含まれます。標準の格納可能クラスには、アカウント、サービス、勘定書、請求書などがあり、さらに Infranet によってあらかじめ定義されたその他のクラスも含まれます。Infranet の機能を拡張するために、新しい格納可能クラスを作成するか、または既存クラスの一部を作成することができます。

格納可能クラスには実際のデータは含まれません。格納可能クラスはオブジェクトの仕様であり、WebSphere Business Integration Adapter ビジネス・オブジェクト定義がビジネス・オブジェクト構造を定義するだけでデータを含まないのと同様です。格納可能クラスには、いくつかのフィールド (整数フィールド、ストリング・フィールドなどの単純フィールド)、配列、または副構造が含まれます。

格納可能クラスがインスタンス化されて実際のデータ値を含むと、格納可能オブジェクトとなります。それぞれの格納可能オブジェクトは、固有の Portal Object ID (POID) によって識別されます。POID には、データベース番号、格納可能クラスの名前、格納可能オブジェクトのインスタンス番号、およびオブジェクト改訂番号が含まれます。

図 4 に、格納可能クラスと格納可能オブジェクトの違いを示します。

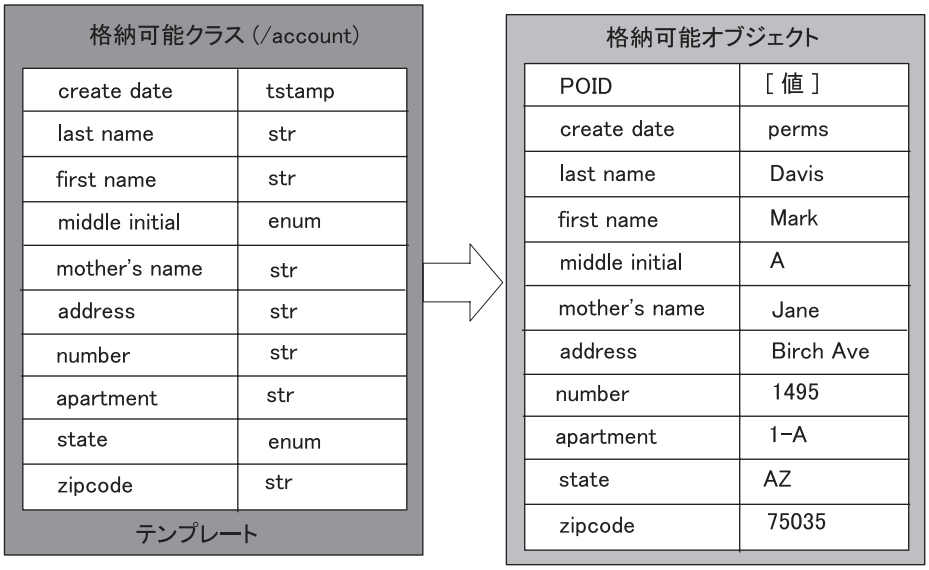


図 4. Infranet の格納可能クラスと格納可能オブジェクト

格納可能クラスは、そのクラスに対して継承および拡張された機能を定義できます。例えば、格納可能クラス /account/email には、account クラス内のすべての情報のほか、email 拡張クラスに特に適用される追加情報が含まれています。したがって、格納可能クラス /account/email は、図 5 に示すように、/account のサブクラスになります。

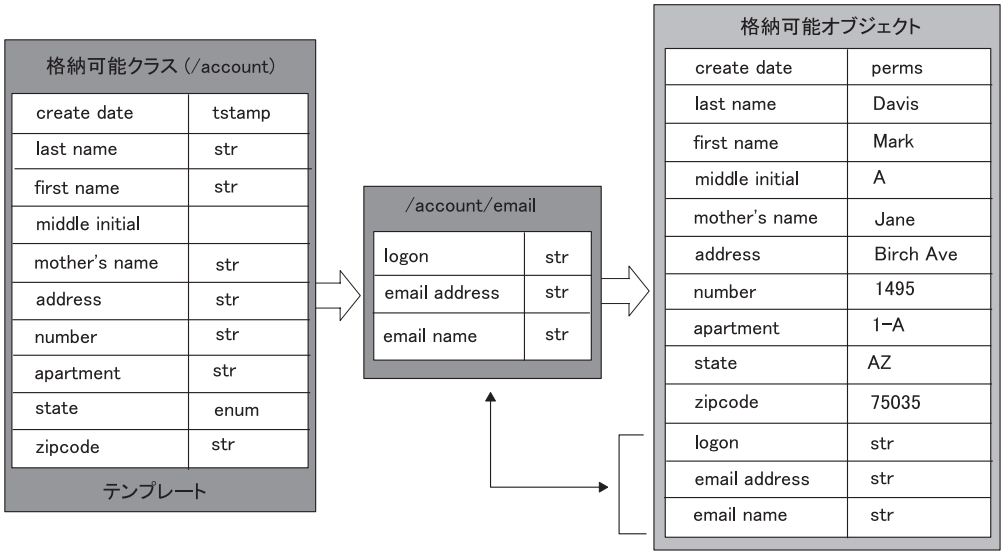


図 5. /account 格納可能クラスの拡張

格納可能オブジェクトは、Infranet アプリケーション・プログラム、スクリプト、およびツール、または、あらゆるカスタム・プログラムおよびプロセスを使用して操作されます。すべてのクライアント・プログラムは、そのタイプにかかわらず、PCM API とプログラミング・ライブラリーを使用して格納可能オブジェクト上で動作します。格納可能オブジェクトは命令コードによって操作されます。命令コードは格納可能オブジェクト上で動作するフィールドのリストを含むルーチンです。

フィールドとフィールド・リスト

フィールドは、Infranet における最も単純なデータ値です。システム内の個々のフィールド名には、固有の ID、名前、タイプ、および定義があります。フィールド名は共有されて、さまざまなクラスおよび命令コードの定義で使用されます。

システムには、新しいフィールドの作成に使用できるフィールド・タイプの基本セットが含まれます。表 5 にフィールド・タイプをリストします。最初の 6 つのタイプは、C などのプログラミング言語のデータ型に対応します。その他のタイプはさらに複雑なデータを含み、それらの値として C 構造体を指すことができます。配列とサブストラクチャーは、他のフィールド・リストを指すためのポインターを持っています。

表 5. Infranet のフィールド・タイプとデータ型

フィールド・タイプ	データ型
PIN_FLDT_INT	符号付き整数
PIN_FLDT_UINT	符号なし整数
PIN_FLDT_ENUM	列挙型整数
PIN_FLDT_NUM	浮動小数点数
PIN_FLDT_TSTAMP	タイム・スタンプ
PIN_FLDT_STR	文字ストリング
PIN_FLDT_BINSTR	バイナリー・ストリング
PIN_FLDT_BUF	任意のサイズのデータ・バッファー
PIN_FLDT_POID	POID
PIN_FLDT_ARRAY	配列
PIN_FLDT_SUBSTRUCT	サブストラクチャー

フィールド・リスト (flist) は、Infranet プログラミング API で使用される基本的なデータ構造です。フィールド・リストはデータ・フィールドと値の対を保持するコンテナーであり、場合によっては他のフィールド・リストを保持することもあります。フィールド・リストは、浮動小数点計算、バッファー、または、メモリーに納まらない大規模データを表すことができます。フィールド・リストは、格納可能オブジェクトと、それらを操作するルーチンまたはプログラムとの間で情報を受け渡します。

1 つの格納可能オブジェクト、例えば /account 格納可能クラスのオブジェクトが、その格納可能クラス仕様を使用するフィールド・リスト (またはその一部) を作成します。フィールド・リストは、それぞれが固有の属性、許可、およびデータ値を持つフィールドのリストです。これらのフィールドは集まって、図 6 に示されるように格納可能オブジェクトの機能を定義します。

フィールド・リストは複数の格納可能オブジェクトを含むことができます。フィールド・リストの構造は、情報がアプリケーションから適切な格納可能オブジェクトに確実に渡されるようになっています。

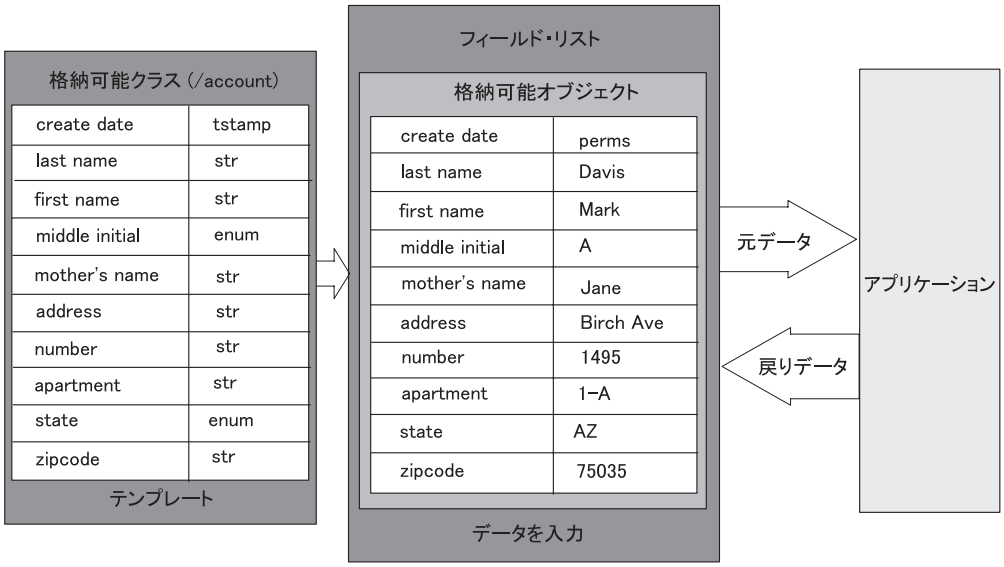


図 6. 格納可能オブジェクトとフィールド・リスト

命令コード

アプリケーションは Infranet システムの命令コードを使用して、格納可能オブジェクト上での操作およびそれらのオブジェクトに含まれるフィールドを実行します。命令コードは Base、Customer Facilities Module (FM)、Activity FM、Billing FM、Terminal FM、および Email FM という機能カテゴリー別にグループ化されて、いくつかの集合を形成します。

オブジェクト上での基本操作には、作成、削除、書き込み、読み取り、および検索があります。その他すべての命令コードは、ロギング・アクティビティー、製品購入に対する勘定請求、クレジット・カード情報のチェック、名前および住所の変更、パスワードの確認、またはアカウンティング・データの記録など、ビジネス・レベル (高水準) のセマンティクスを実装します。これらの高水準の命令コードは、機能モジュールに実装され、そこでは基本命令コードが Storage Manager (SM) に直接実装されます。高水準の各命令コードは、Communication Manager の機能モジュール・ルーチンによってそれぞれ異なる数の基本命令コードに変換されて、Storage Managers に渡されます。

すべてのシステム命令コードは、関連する入出力フィールド・リストを持っています。クライアント・アプリケーションは、関連するイベントがどれかを判断して、適切な命令コードと対応するフィールド・リストを使用して Infranet システムを呼び出し、戻りフィールド・リストとエラー・バッファーを処理します。

メタデータ主導型コネクター

コネクターはメタデータ主導型です。これはつまり、ビジネス・オブジェクト内のメタデータがコネクターの振る舞いを駆動するということです。メタデータとは、ビジネス・オブジェクト内に保管されていて、コネクターがアプリケーションと対話するのを支援するアプリケーションに関するデータです。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・オブジェクトを処理する際に、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を行います。

コネクターは、属性レベルでは Infranet PIN_*FIELDNAME*、動詞レベルでは命令コードの値を使用して、メタデータにより駆動されます。コネクターはメタデータ主導型であるため、コネクター・コードを変更する必要なしに、新規または変更されたビジネス・オブジェクトを処理することができます。ただし、コネクターは、そのビジネス・オブジェクトの以下の点についてさまざまな想定を行います。

- ビジネス・オブジェクトの構造
- 親ビジネス・オブジェクトと子ビジネス・オブジェクトの関係
- アプリケーション固有の情報のフォーマット
- ビジネス・オブジェクトのデータベース表記

したがって、Portal Infranet 用ビジネス・オブジェクトを作成または変更する場合は、コネクターがそれに従うように設計されている規則に準拠して変更を行う必要があります。そうしないと、コネクターは新規または変更されたビジネス・オブジェクトを適切に処理できません。以下のセクションでは、Portal 用ビジネス・オブジェクトの実装についての情報を説明します。

Portal Infranet アプリケーション固有のビジネス・オブジェクト構造

WebSphere Business Integration Adapter ビジネス・オブジェクトは階層構造です。すなわち、親ビジネス・オブジェクトは子ビジネス・オブジェクトを格納し、その子ビジネス・オブジェクトがさらに子ビジネス・オブジェクトを含む、という構造が可能です。親ビジネス・オブジェクトのある属性が 1 つの子オブジェクトを参照すると、カーディナリティーが 1 のコンテナーが出現します。親ビジネス・オブジェクトの属性が子ビジネス・オブジェクトの配列を参照すると、カーディナリティーが n のコンテナー・オブジェクトが出現します。

コネクターは、ビジネス・オブジェクト間におけるカーディナリティー 1 の関係とカーディナリティー n の関係を両方ともサポートします。

WebSphere Business Integration Adapter ビジネス・オブジェクトに対応する Portal Infranet オブジェクト

Infranet には、以下のコンテナー・タイプがあります。

- 格納可能クラス
- 配列
- サブストラクチャー

WebSphere Business Integration Adapter Portal Infranet アプリケーション固有のビジネス ・オブジェクトを定義するにあたっては、オブジェクトが、必要なすべての属性と関係を含む対応する格納可能オブジェクトの Infranet フィールド・リストにマップできるように定義する必要があります。フィールド・リストとビジネス・オブジェクトの関係は、1 対 1 の関係です。

コネクターは処理中に、ビジネス・オブジェクトと Infranet オブジェクトの対応するフィールド・リストとを比較して、構造が一致しない場合は例外をスローします。フィールド・リスト構造の一部となる WebSphere Business Integration Adapter ビジネス・オブジェクトを定義することは可能ですが、変換はサポートされていません。

それぞれの Infranet コンテナー・タイプごとに、アプリケーション固有のビジネス・オブジェクトが必要に応じて作成されます。通常、格納可能クラスはトップレベルのビジネス・オブジェクトになります。**サブストラクチャー・タイプ**のコンテナーはカーディナリティー 1 の子ビジネス・オブジェクトとなる可能性があり、**配列タイプ**のコンテナーはカーディナリティー n の子ビジネス・オブジェクトとなる可能性があります。しかし、サブコンテナーが重要でなく、親の命令コードが子を操作するのに十分である場合は、子ビジネス・オブジェクトは不要です。

図 7 は、WebSphere Business Integration Adapter ビジネス・オブジェクトの構造と Infranet フィールド・リストがどのように対応するかを示しています。Infranet フィールド・リストについては、Portal Infranet 資料を参照してください。

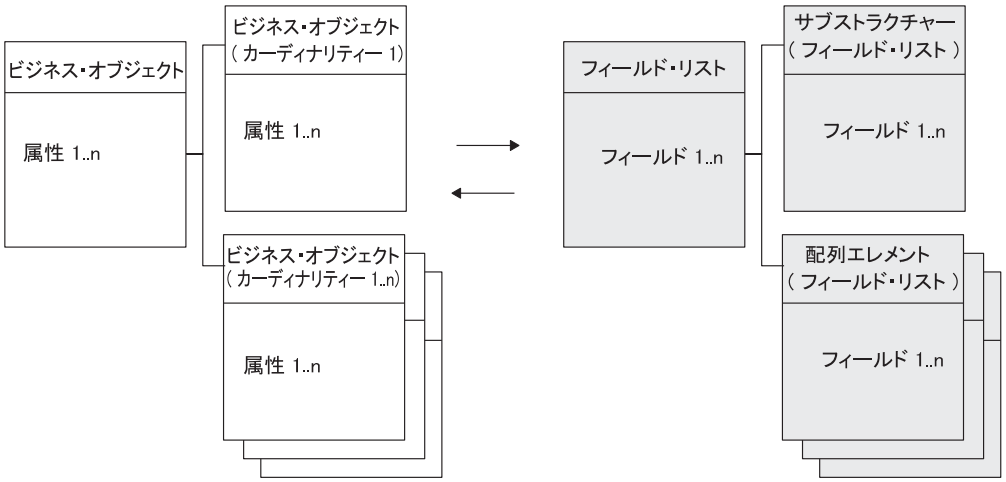


図 7. ビジネス・オブジェクトの構造とフィールド・リスト

図 8 は、Infranet `/account` 格納可能クラスと Portal_Account 階層型ビジネス・オブジェクトの間の対応関係を示しています。格納可能クラスの NameInfo 配列は、トップレベルのビジネス・オブジェクト内でカーディナリティー n の子ビジネス・オブジェクトとなり、Balances サブストラクチャーはカーディナリティー 1 の子ビジネス・オブジェクトとなります。

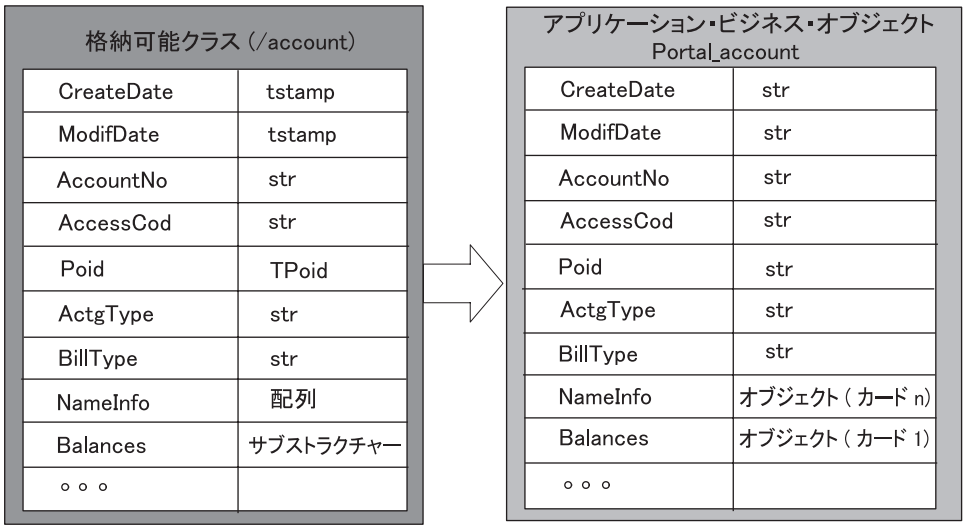


図 8. 格納可能クラスと WebSphere Business Integration Adapter ビジネス・オブジェクトの対応関係の調整

図 9 は、NameInfo 配列と Portal_Contact 子ビジネス・オブジェクトの対応関係を示しています。NameInfo 配列には Phones という名前の配列が含まれ、これは Portal_Contact ビジネス・オブジェクトを親とする子ビジネス・オブジェクトとなります。

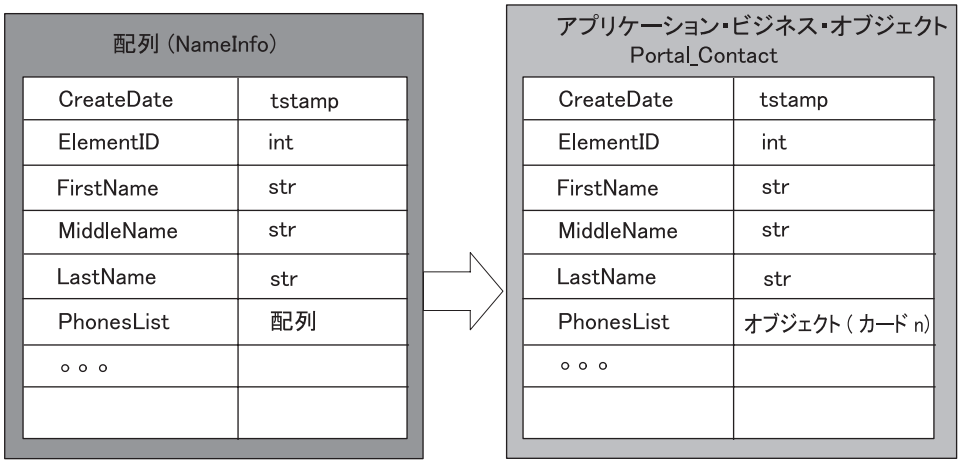


図 9. フィールド・リストの配列と子ビジネス・オブジェクトの対応

一部のフィールド・リストおよび命令コードに限り、特定の属性を必要とする場合があるので注意してください。この場合、追加ユーティリティーのアプリケーション固有ビジネス・オブジェクトを動詞パラメーターとして使用することができます。このオブジェクトはいずれの永続データにも対応せず、フィールド・リストのいくつかの必須フィールドのみを記述します。ユーティリティー・ビジネス・オブジェクトの詳細については、50 ページの『コネクターのユーティリティー・ビジネス・オブジェクト』を参照してください。

ビジネス・オブジェクト属性プロパティー

ビジネス・オブジェクト・アーキテクチャーは、属性に適用され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、コネクターがこのようなプロパティーのいくつかを解釈する方法と、ビジネス・オブジェクトの変更時にそれらを設定する方法を説明します。

Key プロパティー

すべての Portal Infranet 用ビジネス・オブジェクトは、少なくとも 1 つのキー属性を持つ必要があります。キーとなるそれぞれの属性ごとに、Key プロパティーを True に設定してください。

注: 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できません。

トップレベルのビジネス・オブジェクトのキーは、格納可能なオブジェクトの Portal Object ID (POID) です。POID は、各 Infranet データベース・オブジェクトの作成時に割り当てられる固有の 64 ビットの ID です。POID は Infranet オブジェクトの固有キーです。

POID を使用すると、同一の格納可能クラスの複数のインスタンス化が可能になります。例えば、すべてのアカウント・オブジェクトは固有の POID を持っています。POID には、データベース番号、オブジェクト・タイプ、固有 ID、および改訂番号の 4 つのコンポーネントが含まれます。

子ビジネス・オブジェクトのキー値

Infranet の配列はそのエレメント ID によって識別され、エレメント ID はその配列の固有キーとなっています。配列は通常は子ビジネス・オブジェクトに対応するので、Portal Infranet 階層型ビジネス・オブジェクトでは、子ビジネス・オブジェクトのキーが配列エレメント ID と親 POID を指定します。

一般的な規則として、2 次レベルの子ビジネス・オブジェクトはそのエレメント ID の属性のみを必要とします。Create または Update の操作中には、すべての ElementId と POID の値がビジネス・オブジェクト内に含まれます。

Foreign key プロパティー

このプロパティーは、子ビジネス・オブジェクトが親ビジネス・オブジェクトに関連付けるために使用します。動詞処理中に、子を別個に実行する場合は、外部キー・フィールドを親ビジネス・オブジェクトから設定します。使用する親ビジネス・オブジェクトからの属性の名前は、外部キー属性のアプリケーション固有情報で指定します。

Required プロパティー

コネクターは Required プロパティーを使用しません。

Max length プロパティー

Max Length プロパティーは 255 に設定してください。

Default value プロパティー

指定すると、コネクターは属性のデフォルト値を使用します。

ビジネス・オブジェクトの定義に関するガイドライン

Portal Infranet アプリケーション固有のビジネス・オブジェクトを定義する際には、以下のガイドラインを使用してください。

- オブジェクトの POID 属性が、オブジェクト定義の 1 番目の属性となること。
- オブジェクトのその他の ID 属性は、オブジェクト定義の POID 属性の後に続くこと。
- キー属性および外部キー属性は ID 属性の後に続くこと。
- 残りの属性は、参照されるオブジェクトおよび含まれているオブジェクトを除いて、キー属性の後に続き、論理的にソートされること。
- すべての参照されるオブジェクト (カーディナリティー 1 のもの) または含まれているオブジェクト (カーディナリティー n のもの) は、その後に続くこと。
- ObjectEventId が、オブジェクト定義の最後の属性となること。

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から、コネクターは、ビジネス・オブジェクトの処理方法に関するアプリケーション依存命令を取得します。このメタデータは、ビジネス・オブジェクトの属性のプロパティーおよび構造とともに使用されます。Portal Infranet アプリケーション固有のビジネス・オブジェクトを作成する際には、ビジネス・オブジェクト定義のアプリケーション固有情報を、コネクターの要求する構文と一致させる必要があります。

このセクションでは、Portal 用ビジネス・オブジェクトのオブジェクト、属性、および動詞レベル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。

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・レベルでは、アプリケーション固有情報は以下のような Infranet エンティティーを記述します。

- 格納可能クラスの場合、ビジネス・オブジェクトのアプリケーション固有情報は、Create または Update 動詞の入力フィールド・リストがビジネス・オブジェクトの構造と等しい場合、格納可能クラスの名前を指定します。
- ビジネス・オブジェクトがフィールド・リストの配列またはサブストラクチャーに対応する子オブジェクトである場合、オブジェクトのアプリケーション固有情報には、Infranet フィールド名が含まれます。

例えば、Portal_Account ビジネス・オブジェクトは、`CN=/account` 格納可能クラスをオブジェクト・レベルのアプリケーション固有情報として指定し、Portal_BillInfo 子ビジネス・オブジェクトは、`FN=PIN_FLD_BILLINFO` フィールドをオブジェクト・レベルのアプリケーション固有情報として指定します。

属性レベルのアプリケーション固有情報

属性レベルでは、アプリケーション固有情報を使用し、特定の命令コードの実行に使用するフィールド・リスト構造を作成します。アプリケーション固有情報は名前と値の組のリストです。この構造により、ユーティリティー・ビジネス・オブジェクトの使用によって以前に課されたが、命令コードのフィールド・リスト構造の定義に不要となった制約を取り外すことができます。属性のアプリケーション固有情報のフォーマットは以下のとおりです。

```
FN=FIN_FLD_POID;Create=true;Update=false;Delete=true;
Retrieve=true;CreateT=;UpdateT=PIN_FL_NAMEINFO;
DeleteT=;RetrieveT=;O=false;CreateO=;UpdateO=;
DeleteO=;RetrieveO=;ParentAtt=;Alone=false
```

古いビジネス・オブジェクトから新しいビジネス・オブジェクトへの変換

属性レベルのアプリケーション固有情報に関する上記のフォーマットは、コネクター・バージョン 4.0.x との後方互換性のためにサポートされています。しかし、将来のリリースでは後方互換性はサポートされないため、PortalODA を使用して古いビジネス・オブジェクト定義を新しいビジネス・オブジェクト定義に変換する必要があります。

PortalODA を使用して古いビジネス・オブジェクト定義を新しいビジネス・オブジェクト定義に変換するには、以下の手順を行います。

1. 子ビジネス・オブジェクトから親ビジネス・オブジェクトにリンクする子ビジネス・オブジェクトの外部キー・フィールドにマークを付けます。アプリケーション固有情報には ParentAtt というタグがあります。このタグの値を、外部キーに使用する必要がある親ビジネス・オブジェクトからの属性の名前に設定します。
2. 必要な場合は、タイプ「object」の属性に特定の動詞タグでマークを付けます。CreatT、UpdateT、DeleteT、および RetrieveT については、表 6 を参照してください。

表 6 に、属性のアプリケーション固有情報のフォーマットを示します。

表 6. 属性に関するアプリケーション固有情報

名前	説明	可能な値	デフォルト値
FN	Infranet のフィールド名を表すフィールド名。		
Create	属性が Create 動詞のフィールド・リストの一部であるかどうかを示します。	true または false	false
Update	属性が Update 動詞のフィールド・リストの一部であるかどうかを示します。	true または false	false
Delete	属性が Delete 動詞のフィールド・リストの一部であるかどうかを示します。	true または false	false
Retrieve	属性が Retrieve 動詞のフィールド・リストの一部であるかどうかを示します。	true または false	false
CreateT	Create 動詞のフィールド・リストの作成時に属性のコンテナーとして機能する Infranet フィールド名。		null

表 6. 属性に関するアプリケーション固有情報 (続き)

名前	説明	可能な値	デフォルト値
UpdateT	Update 動詞のフィールド・リストの作成時に属性のコンテナーとして機能する Infranet フィールド名。		null
DeleteT	Delete 動詞のフィールド・リストの作成時に属性のコンテナーとして機能する Infranet フィールド名。		null
RetrieveT	Retrieve 動詞のフィールド・リストの作成時に属性のコンテナーとして機能する Infranet フィールド名。		null
O	応答ビジネス・オブジェクトで属性を更新する必要があるかどうかを示します。	true または false	false
CreateO	Create 動詞処理用の応答ビジネス・オブジェクトを更新するために使用する属性の値を表すフィールドを持つ Infranet フィールド名を示します。例えば、FN=PIN_FLD_POID および CreateO=PIN_FLD_NAMEINFO の場合は、コネクターはフィールド PIN_FLD_POID を検索し、この属性の応答ビジネス・オブジェクトにこのフィールドの値を設定します。 Main flist からフィールドを更新する必要がある場合は、値に Main をセットします。		Main
UpdateO	Update 動詞処理用の応答ビジネス・オブジェクトを更新するために使用する属性の値を表すフィールドを持つ Infranet フィールド名を示します。 Main flist からフィールドを更新する必要がある場合は、値に Main をセットします。		Main
DeleteO	Delete 動詞処理用の応答ビジネス・オブジェクトを削除するために使用する属性の値を表すフィールドを持つ Infranet フィールド名を示します。 Main flist からフィールドを更新する必要がある場合は、値に Main をセットします。		Main
RetrieveO	Retrieve 動詞処理用の応答ビジネス・オブジェクトを検索するために使用する属性の値を表すフィールドを持つ Infranet フィールド名を示します。 Main flist からフィールドを更新する必要がある場合は、値に Main をセットします。		Main

表 6. 属性に関するアプリケーション固有情報 (続き)

名前	説明	可能な値	デフォルト値
ParentAtt	このフィールドは、子ビジネス・オブジェクトがキー・フィールドの設定に使用する親ビジネス・オブジェクトの属性を定義するために使用します。これらのフィールドは、子ビジネス・オブジェクトで外部キー・フィールドとしてマークする必要があります。		NULL
Alone	このフィールドは、子ビジネス・オブジェクトを親ビジネス・オブジェクトの一部として実行するのではなく、別個に実行することを表すために子ビジネス・オブジェクトが使用します。	true または false	false

図 10 は Portal_Account ビジネス・オブジェクトを示し、ビジネス・オブジェクトおよび属性に応じたアプリケーション固有情報を示しています。

図 10. ビジネス・オブジェクトおよび属性のアプリケーション固有情報

動詞のアプリケーション固有情報のフォーマット

Portal Infranet 用ビジネス・オブジェクトの動詞レベルのアプリケーション固有情報は、ビジネス・オブジェクトおよび動詞が実行するアクションに応じた固有の Infranet 命令コードを指定する必要があります。

命令コードは、Infranet アプリケーションとそのデータベースの間でデータを受け渡し、格納可能オブジェクトでの操作を実行します。Infranet オブジェクト上でのそれぞれのアクションは、特定の命令コードを持っています。

命令コードは、Infranet 格納可能オブジェクトを、フィールド名と値の対のリストであるフィールド・リストの形式で渡します。各命令コードは、特定の入力および出力フィールド・リストを持っています。コネクターは、ビジネス・オブジェクト要求を、命令コードが要求する入力フィールド・リストに変換する必要があります。ビジネス・オブジェクト要求の実行時には、コネクターは以下の基本ステップを実行します。

1. ビジネス・オブジェクト・インスタンス内の値とビジネス・オブジェクト定義の情報を使用して、入力フィールド・リストを作成します。
2. 入力フィールド・リストを引き数として、Portal Infranet 命令コードを実行します。
3. 命令コードは出力フィールド・リストを戻し、コネクターは出力フィールド・リストの情報でビジネス・オブジェクトを更新します。

動詞のアプリケーション固有情報と、場合によっては追加のユーティリティー・ビジネス・オブジェクトを使用すると、コネクターは命令コードごとに適切なフィールド・リストを生成することができます。ユーティリティー・ビジネス・オブジェクトを使用すると、コネクターは、適切な入力フィールド・リストを作成したり、ビジネス・オブジェクトを出力フィールド・リストで適切に更新したりすることができます。ユーティリティー・ビジネス・オブジェクトは、トップレベルのビジネス・オブジェクトのビジネス・オブジェクト定義の一部として提供されます。属性レベルのアプリケーション固有情報の新しい定義により、ユーティリティー・ビジネス・オブジェクトが冗長になります。ユーティリティー・ビジネス・オブジェクトの機能は、後方互換性のためにコネクター・バージョン 4.0.x でサポートされていますが、将来のリリースでは廃止される可能性があります。

動詞のアプリケーション固有情報の構文

動詞のアプリケーション固有情報の必須構文は、以下のとおりです。

```
<opcode>['#'<flag>][';'transaction
enabled][';'<input_flist_model>][';'<output_flist_model>]
```

ここで、以下のように説明されます。

opcode	名前の先頭に PCM_OP_ のない Infranet 命令コード。すなわち opcode は、キーワード ERROR の可能性があります。キーワード ERROR は Infranet 制約を示します。コネクターは、操作が発生した場合にエラーを提示します。これは、子レベルでの操作の禁止を示すことがあります。例えば、Portal_BillInfo ビジネス・オブジェクトは単独では削除できません。親オブジェクトと一緒に削除する必要があります。
flag	命令コードのオプションの引き数。命令コードのフラグについては、Portal Infranet 資料を参照してください。
transaction enabled	Portal Infranet の命令コードの中には、命令コードの機能のクリティカルな性質のために固有のトランザクションを保持しているものがあります。このフラグは、そのようなトランザクションの有無を Portal Infranet コネクターに伝えるために使用されます。値「true」は、命令コードが固有のトランザクションを保持していることを示します。Portal Infranet コネクターのデフォルトでは、このプロパティーの値は「false」になっています。
input_flist_model	入力フィールド・リストのユーティリティー・ビジネス・オブジェクトの名前といくつかのオプションのパラメーター、またはキーワードのどちらか。構文は、<input_flist_model>[#<parameter>] です。可能なキーワードは、NotNull、NORMAL、または OnlyPoid です。これらのキーワードは以下のように定義されます。 • NORMAL は、現行のビジネス・オブジェクトがモデルであることを示します。すなわち、コネクターが現行ビジネス・オブジェクトを入力フィールド・リストに直接変換できるということを示します。 • OnlyPoid は、現行の命令コードがビジネス・オブジェクトから POID のみを必要とすることを意味します。この結果、コネクターは処理を単純化できます。 • NotNull は、配列をヌルに設定することによってこのオブジェクトを削除してはいけないということを示します。 ユーティリティー・ビジネス・オブジェクトについては、50 ページの『コネクターのユーティリティー・ビジネス・オブジェクト』を参照してください。
output_flist_ model	出力フィールド・リストのユーティリティー・ビジネス・オブジェクトの名前といくつかのオプションのパラメーター、またはキーワードのどちらか。構文は、<output_flist_model>[#<parameter>] です。可能なキーワードは、NoRewrite および Flat です。これらのキーワードは以下のように定義されます。 • NoRewrite は、命令コードによって戻される出力フィールド・リストを使用してビジネス・オブジェクトを上書きしてはいけないことを示します。 • Flat は、コネクターがトップレベルのビジネス・オブジェクトから子ビジネス・オブジェクトの属性を取得する必要があることを示します。 ユーティリティー・ビジネス・オブジェクトについては、50 ページの『コネクターのユーティリティー・ビジネス・オブジェクト』を参照してください。

命令コード・アプリケーションの規則

Create、Update、および Delete 操作の格納可能クラスのレベルには、1 つ以上の命令コードが存在します。しかし、コネクターはそれぞれの動詞に対して命令コードを 1 つしかサポートしません。したがって、動詞ごとに、ビジネス・オブジェクトおよび動詞の操作に最も適した命令コードを選択する必要があります。

親レベルと子レベルで、必要な命令コードが異なる場合があります。子オブジェクト用の命令コードが存在するときは、Portal Infranet は親オブジェクトの命令コードではなくそちらを使用するように勧めます。Infranet のサブコンポーネントを更新するために特定の命令コードが必要な場合は、その子オブジェクト (サブコンポーネント) のために新しいアプリケーション固有ビジネス・オブジェクトを作成して、動詞のアプリケーション固有情報内に命令コードを指定する必要があります。

コネクターが階層型ビジネス・オブジェクト用の入力フィールド・リストを作成する際に、子ビジネス・オブジェクト動詞が親と同じ命令コードを持っている場合、コネクターは子を親と同じフィールド・リストに入れます。そうでない場合、コネクターは子オブジェクト用に別のフィールド・リストを作成します。Infranet はフィールド・リスト内のレベルを使用して、配列とサブストラクチャーを指定します。コネクターは入力フィールド・リストの作成を開始する際に、レベルをゼロに設定します。ビジネス・オブジェクトに子が存在する場合、子オブジェクトを処理するたびにレベルが 1 ずつ増分します。階層の一部として実行するのではなく、単独で実行する場合にビジネス・オブジェクトで異なる命令コードが必要な場合は、同様の構造を持つ異なるビジネス・オブジェクトを使用してください。

Create 操作の場合、親の命令コードが子の命令コードの前に実行されます。Delete 操作の場合、子の命令コードが親の命令コードの前に実行されます。Update および Retrieve 操作の場合は、必須の実行順序はありません。

すべての格納可能クラスは、命令コード READ_OBJ とルート・オブジェクトの POID を使用して検索することができます。

コネクターのユーティリティー・ビジネス・オブジェクト

それぞれの Infranet 命令コードは、特定の入力フィールド・リストを必要とし、特定の出力フィールド・リストを戻します。コネクターをメタデータ主導型にするには、ビジネス・オブジェクト・インスタンスおよび動詞を適切なフィールド・リストに変換するために必要なフィールドを、ビジネス・オブジェクトからコネクターに対して提供する必要があります。命令コードのフィールド・リストはそれぞれ異なるので、コネクターがすべての入出力フィールド・リストのために必要とする全情報を提供する単一の Portal Infranet 用ビジネス・オブジェクトを作成することはおそらく不可能です。その代わりに、アプリケーション固有のビジネス・オブジェクト定義を補足する特別なユーティリティー・ビジネス・オブジェクト定義が必要な場合があります。

注: コネクター・バージョン 4.0.x の属性レベルのアプリケーション固有情報の定義は、ユーティリティー・ビジネス・オブジェクトを必要としません。コネクター・バージョン 4.0.x は後方互換性のために属性レベルのアプリケーション固有情報もサポートしていますが、将来のリリースでは、PortalODA を使用して古いビジネス・オブジェクト定義を新しいビジネス・オブジェクト定義に変

換する必要があります。手順については、 44 ページの『古いビジネス・オブジェクトから新しいビジネス・オブジェクトへの変換』を参照してください。

ユーティリティー・ビジネス・オブジェクト定義は、統合ブローカーによって送信されるビジネス・オブジェクト・インスタンスにはなりません。コネクターは単にそれらの定義を使用して、特定の命令コードのために必要な入出力フィールド・リストを構成するだけです。ユーティリティー・ビジネス・オブジェクト定義は、アプリケーション固有のビジネス・オブジェクトの設計中に設計および作成する必要があり、また、すべてのユーティリティー・ビジネス・オブジェクトとすべてのアプリケーション固有ビジネス・オブジェクトを両方ともサポートするようにコネクターを定義する必要があります。ユーティリティー・ビジネス・オブジェクトの例については、 52 ページの『ユーティリティー・ビジネス・オブジェクトの例: Create 動詞』を参照してください。

ユーティリティー・ビジネス・オブジェクトが必要かどうかを判断するには、Infranet 格納可能クラスと、動詞の操作の実行に使用される命令コードの入出力フィールド・リストを確認してください。

Portal Infranet ユーティリティー・ビジネス・オブジェクトは、Portal 用ビジネス・オブジェクトとはフォーマットの異なるアプリケーション固有情報を使用します。以下のセクションでは、このフォーマットについて説明します。

ユーティリティー・ビジネス・オブジェクトのアプリケーション固有情報

ユーティリティー・ビジネス・オブジェクト内の属性のアプリケーション固有情報は、フィールド・リストに追加すべきフィールドを指定し、さらにフィールド・リストのフィールドに使用するための値も含みます。ユーティリティー・ビジネス・オブジェクトでは、単純属性およびコンテナー属性用に、属性のアプリケーション固有情報を以下のように定義する必要があります。

- 単純属性の場合は、フィールド・リストにフィールド名を指定して、そのフィールドに値を定義してください。フィールドは、ビジネス・オブジェクト・インスタンス内の対応する属性から抽出するか、またはアプリケーション固有情報内に提供されているデフォルト値を使用できます。この記述の構文は以下のとおりです。

  ```
  <flist_fieldname>[:[<bus_object_attributename>]:<default_value>]
  ```

- 配列または構造属性の場合、アプリケーション固有情報には、フィールド・リストから除外する必要のあるアプリケーション固有ビジネス・オブジェクト内の属性のリストが含まれています。属性は、コロンで区切られます。

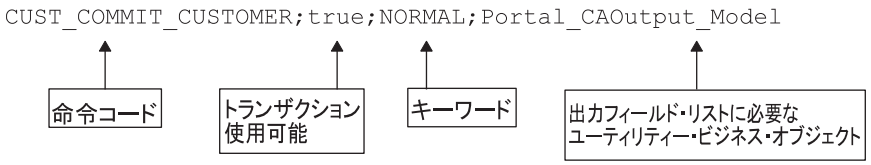
  ```
  [< bus_object_attributename>]([:< bus_object_attributename>])*
  ```

動詞の記述はユーティリティー・オブジェクトには使用されません。

以降のセクションでは、Create、Update、Retrieve、および Delete の各動詞に応じたアプリケーション固有情報の例について説明します。これらの例では、Portal_Account 階層型ビジネス・オブジェクトを使用して、動詞のアプリケーション固有情報の性質を具体的に示します。

ユーティリティー・ビジネス・オブジェクトの例: Create 動詞

一例として、Portal_Account トップレベル・ビジネス・オブジェクト内の Create 動詞のための動詞のアプリケーション固有情報について考えてみます。



CUST_COMMIT_CUSTOMER は、コネクターが新しいカスタマー・アカウント (/account 格納可能オブジェクト) を作成するために使用する命令コードです。命令コードは独自のトランザクションを持っているので、「使用可能トランザクション」は「true」に設定されています。input_flist_model フィールドのキーワード NORMAL は、コネクターが直接対応することを示します。すなわち、ビジネス・オブジェクトが入力フィールド・リストに必要なすべてのフィールドを提供するので、コネクターは入力フィールド・リストを作成するために補足情報を必要としないということです。

Create Account 操作の CUST_COMMIT_CUSTOMER 命令コードは、PIN_FLD_ACCOUNT_OBJ フィールドに新しいカスタマーの ID を含む出力フィールド・リストを戻します。この ID の値は WebSphere Business Integration Adapter システムに戻される必要があります。コネクターが新しい ID を取得できるように、Business Object Designer は Portal_C[reate]A[ccount]Output_Model ユーティリティー・ビジネス・オブジェクト定義を作成しています。アプリケーション固有情報の output_flist_model フィールドは、命令コードによって戻される出力フィールド・リストを、コネクターが読み取るために使用するユーティリティー・ビジネス・オブジェクトとして、Portal_CAOutput_Model を指定します。

Portal_CAOutput_Model ユーティリティー・オブジェクトは、1 つの属性 Poid を含みます。この属性のアプリケーション固有情報はコネクターに対して、戻りフィールド・リストから PIN_FLD_ACCOUNT_OBJ の値を抽出して新しいカスタマーの Portal Infranet Object ID を取得するように指示します。コネクターはこの値を、統合ブローカーに戻すビジネス・オブジェクトに挿入します。ユーティリティー・ビジネス・オブジェクトを 図 11 に示します。

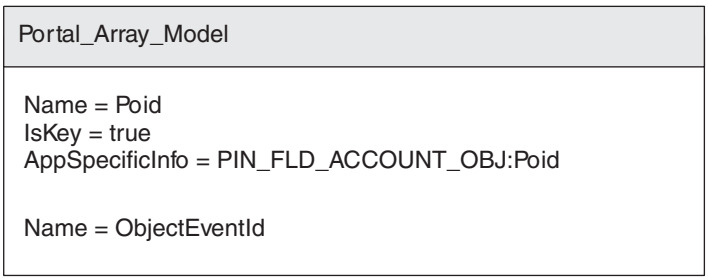


図 11. Portal_CAOutput_Model ユーティリティー・ビジネス・オブジェクトの定義

Create 操作用のフィールド・リストの作成: Portal_Account ビジネス・オブジェクトは、図 12 のような階層型ビジネス・オブジェクトです。

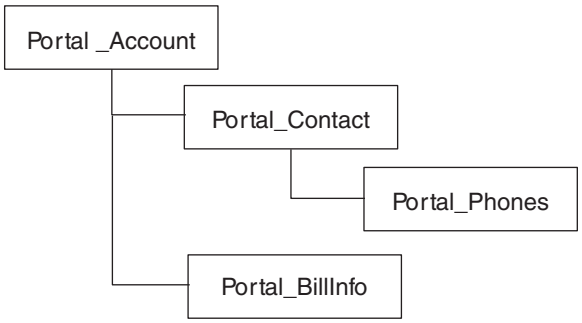


図 12. Portal_Account 階層型ビジネス・オブジェクトの図

Create 操作では、コネクターは子ビジネス・オブジェクトの動詞のアプリケーション固有情報を調べて、命令コードが親ビジネス・オブジェクトによって使用されるものと同じかどうかを判別します。Portal_Account ビジネス・オブジェクトの場合、親ビジネス・オブジェクトと子ビジネス・オブジェクトの命令コードは同じであり、コネクターは、ビジネス・オブジェクト全体の Create 操作のために単一のフィールド・リストを作成します。図 13 は、コネクターが Infranet に対して Create 呼び出しを行う場合に使用する単一の命令コードとフィールド・リストを示しています。

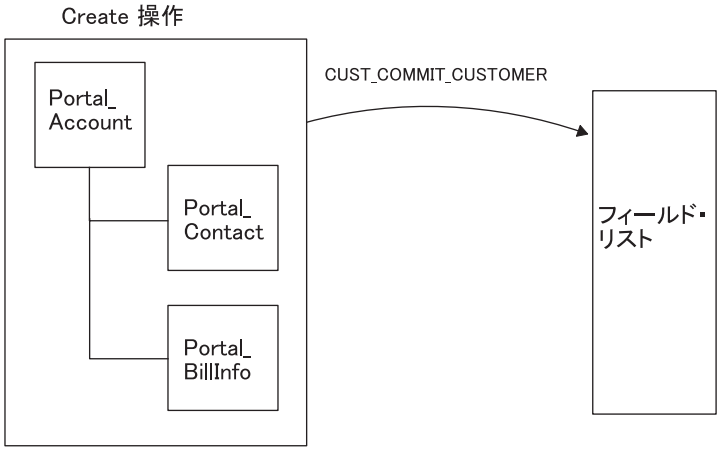


図 13. Portal_Account 階層型ビジネス・オブジェクトの Create 操作用フィールド・リスト

ただし、コネクターは子ビジネス・オブジェクトの配列を含むフィールド・リストを作成しなければならないので注意してください。したがって、動詞のアプリケーション固有情報には、その配列が出現するべきフィールド・リスト内のレベルを示すフィールドが含まれる必要があります。例えば、Portal_Contact 子ビジネス・オブジェクトの Create 動詞のアプリケーション固有情報は以下のとおりです。

```
CUST_COMMIT_CUSTOMER
```

Portal_Phone 子ビジネス・オブジェクトの Create 動詞のアプリケーション固有情報は以下のとおりです。

```
CUST_COMMIT_CUSTOMER
```

ユーティリティー・ビジネス・オブジェクトの例: Update 動詞

この例は、Portal_Account トップレベル・ビジネス・オブジェクト内の Update 動詞のための動詞のアプリケーション固有情報を示します。

```
CUST_SET_STATUS;false;Portal_Array_Model#PIN_FLD_STATUSES;NoRewrite
```

CUST_SET_STATUS は、アカウント・オブジェクトの更新に必要な命令コードです。命令コードは独自のトランザクションを持っていないので、「使用可能トランザクション」は「false」に設定されています。この動詞の操作では、コネクターをビジネス・オブジェクト・インスタンスからフィールド・リストに直接対応させることができません。なぜなら、Portal_Account ビジネス・オブジェクトは入力フィールド・リストに必要なすべての情報を提供しないからです。コネクターはフィールド・リストを作成するために追加情報を必要とするので、input_flist_model フィールドは、コネクターが入力フィールド・リストを構成するために使用するユーティリティー・ビジネス・オブジェクト定義を指定します。このユーティリティー・オブジェクトは Portal_Array_Model と命名されます。

output_flist_model フィールドにはキーワード NoRewrite が含まれます。このキーワードは、命令コードによって戻される出力フィールド・リストを使用してビジネス・オブジェクトを上書きしてはいけないことを示します。

Portal_Array_Model ユーティリティー・ビジネス・オブジェクト:

Portal_Array_Model は、図 14 に示されるような階層型ビジネス・オブジェクト定義です。Portal_Array_Model には、Update 操作の命令コードのための入力フィールド・リストをコネクターが作成するために必要な情報が含まれます。特に、入力フィールド・リストには、Portal_Account ビジネス・オブジェクト定義に含まれていない配列が必要です。Portal_Array_Model ユーティリティー・オブジェクトを使用すると、コネクターは配列を作成できます。

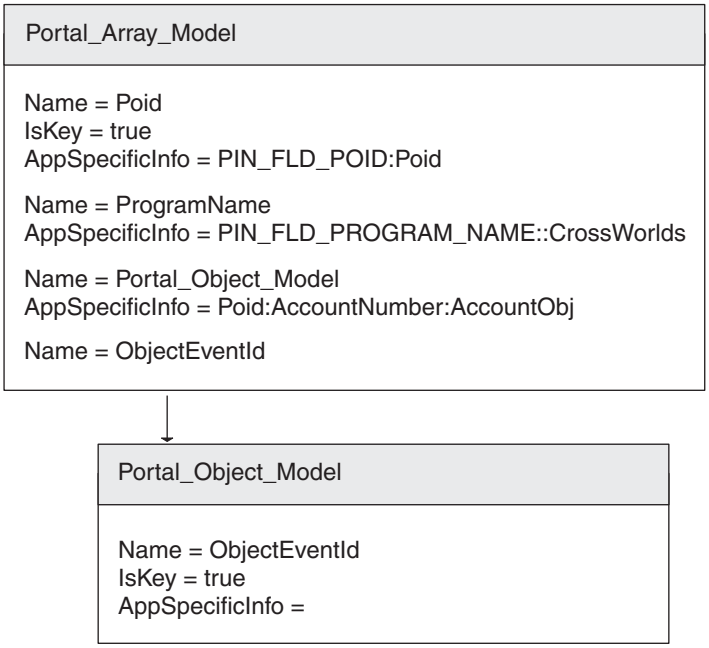


図 14. Portal_Array_Model ユーティリティー・ビジネス・オブジェクトの定義

前に述べましたが、Portal_Account Update 動詞のアプリケーション固有情報の input_flist_model フィールドには、以下のテキストが含まれています。

```
Portal_Array_Model#PIN_FLD_STATUSES
```

このテキストは、フィールド・リストを作成するために使用されるユーティリティー・ビジネス・オブジェクトを示し、コネクターが入力フィールド・リストに配置する必要がある配列の名前を指定します。コネクターは実行時には、ユーティリティー・ビジネス・オブジェクト定義と Portal_Account アプリケーション固有のビジネス・オブジェクト定義の両方を使用して、入力フィールド・リストを作成します。

コネクターは、以下の手順で入力フィールド・リストを作成します。

1. コネクターは、フィールド・リストの作成を PIN_FLD_POID フィールドから開始し、ビジネス・オブジェクト・インスタンスから POID の値を取得します。
2. コネクターは、PIN_FLD_PROGRAM_NAME 用のフィールドをフィールド・リストに追加します。Portal_Account ビジネス・オブジェクト定義にはこの属性が含まれないため、ビジネス・オブジェクト・インスタンス内には対応する値が存在しません。したがって、値はアプリケーション固有情報内にストリング CrossWorlds として定義されます。
3. コネクターは PIN_FLD_STATUSES という配列をフィールド・リストに追加します。命令コード PCM_OP_CUST_SET_STATUS のためのフィールド・リストには PIN_FLD_STATUSES 用の配列が必要であるため、Portal_Array_Model はコンテナー属性を組み込んで、コネクターがフィールド・リスト内に配列を作成できるようにする必要があります。コネクターは、Update 動詞のアプリケーション固有情報に示されたとおりの名前を配列に付けます。

   コネクターは、現行ビジネス・オブジェクトである Portal_Account を配列のモデルとして使用します。すなわち、コネクターは、現行ビジネス・オブジェクト内に指定されているフィールドをフィールド・リストの配列に挿入します。フィールドの中には不要なものもあるので、ユーティリティー・ビジネス・オブジェクト内のコンテナー属性のアプリケーション固有情報が、無視するべき属性を指定します。コンテナー属性 Portal_Object_Model は、以下の属性を指定します。

   ```
   Poid:AccountNumber:AccountObj
   ```

   したがって、コネクターは配列を構成するために、Portal_Account のビジネス・オブジェクト定義を調べ、POID、AccountNumber、および AccountObj 属性を無視し、同ビジネス・オブジェクト定義内の残りの属性、すなわち PIN_FLD_STATUS と PIN_FLD_STATUS_FLAGS のみを使用して配列を作成します。

結果として、Update 動詞の入力フィールド・リストは以下のようになります。

```
PIN_FLD_POID              POID    <value from bus obj instance>
PIN_FLD_PROGRAM_NAME      STR     "CrossWorlds"
PIN_FLD_STATUSES          ARRAY
   PIN_FLD_STATUS         ENUM    <value from bus obj instance>
   PIN_FLD_STATUS_FLAGS     INT      <value from bus obj instance>
```

このフィールド・リストには、命令コード PCM_OP_CUST_SET_STATUS の入力フィールド・リストのための必須フィールドが含まれます。

子ビジネス・オブジェクトの処理: Update 操作では、アカウント格納可能オブジェクトの更新、アカウント格納可能オブジェクトのカスタマー連絡先情報の更新、および、アカウント格納可能オブジェクトの請求情報の更新のどれを行うかによって、必要な Infranet 命令コードは異なります。

したがって、カスタマー連絡先情報とカスタマー請求情報は同じ格納可能クラスの一部であるにもかかわらず、コネクターは別々の命令コードを使用して、Portal_Account トップレベル・ビジネス・オブジェクトと Portal_Contact および Portal_BillInfo の子ビジネス・オブジェクトを更新する必要があります。さらに、コネクターは、それぞれの命令コードごとに別々の入力フィールド・リストを生成する必要があります。図 15 は、Portal_Account 階層型ビジネス・オブジェクトの更新に必要なフィールド・リスト群を示しています。

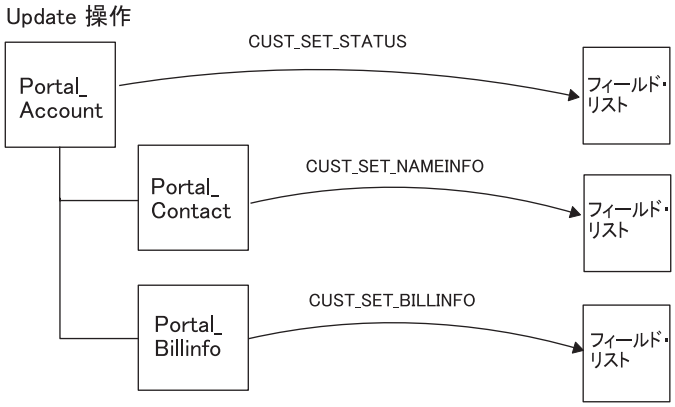


図 15. Portal_Account 階層型ビジネス・オブジェクトの Update 操作用フィールド・リスト群

ユーティリティー・ビジネス・オブジェクトの例: Retrieve 動詞

この例は、Portal_Account トップレベル・ビジネス・オブジェクト内の Retrieve 動詞のアプリケーション固有情報を示します。

```
READ_OBJ;false;Only_Poid;Flat#Portal_BillInfo
```

コネクターは、READ_OBJ 命令コードを使用して、データベースから格納可能オブジェクトを読み取ります。この命令コードはトランザクションを持っていないので、「使用可能トランザクション」は「false」に設定されています。通常、この命令コードは、すべての Retrieve 操作に使用できます。

input_flist_model フィールドは、現行の命令コードがビジネス・オブジェクトから POID のみを必要とすることを示します。その他のフィールドは入力フィールド・リストには必要ありません。

命令コードは、オブジェクトの POID を戻します。オブジェクト内のその他すべてのフィールド (すべての配列エレメントを含みます) が、戻りフィールド・リストの POID の後に追加されます。output_flist_model フィールドには、キーワード Flat とともにパラメーター Portal_BillInfo が含まれます。これは、コネクターが Portal_BillInfo 子ビジネス・オブジェクトを作成するために必要な情報が、配列内ではなく flat フィールド・リストに含まれることを意味します。

ユーティリティー・ビジネス・オブジェクトの例: Delete 動詞

最後の例は、Portal_Account トップレベル・ビジネス・オブジェクト内の Delete 動詞のアプリケーション固有情報です。

```
CUST_DELETE_ACCT;false;Only_Poid;NoRewrite
```

コネクターは、CUST_DELETE_ACCT 命令コードを使用して、データベースから格納可能オブジェクトを削除します。この命令コードは独自のトランザクションを持っていないので、「使用可能トランザクション」は「false」に設定されています。input_flist_model フィールドは、現行の命令コードがビジネス・オブジェクトから POID のみを必要とすることを示します。その他のフィールドは入力フィールド・リストには必要ありません。output_flist_model フィールドにはキーワード NoRewrite が含まれます。このキーワードは、命令コードによって戻される出力フィールド・リストを使用してビジネス・オブジェクトを上書きしてはいけないことを示します。

Infranet は論理削除方式のアプリケーションなので注意してください。いくつかのオブジェクトでは、削除操作は状況の変更となります。例えば、Portal_Service ビジネス・オブジェクトの場合、Delete 動詞のアプリケーション固有情報は以下のようになります。

```
CUST_SET_STATUS;Portal_DSInput_Model#PIN_FLD_STATUSES#NotNull
```

このテキストは、論理的な Delete 操作の命令コードを CUST_SET_STATUS と指定し、input_flist_model は、ユーティリティー・オブジェクト Portal_D[elete]S[ervice]Input_Model であると指定します。このユーティリティー・オブジェクトは、ヌルに設定されない PIN_FLD_STATUSES 配列を含むフィールド・リストを定義します。

コンテキスト主導型動詞の振る舞い

コネクターは、階層型ビジネス・オブジェクトと単一の子ビジネス・オブジェクトの両方を処理できなくてはならないので (例えば、Portal_Contact ビジネス・オブジェクトは親オブジェクトがなくても送信可能)、一部のメタデータ主導型の決定は、ビジネス・オブジェクトおよび動詞のコンテキストによって行われます。動詞に応じて、ビジネス・オブジェクト階層全体に対して 1 つの固有の命令コード、または、各ビジネス・オブジェクトごとに 1 つの命令コードを指定する必要があります。このために、各ビジネス・オブジェクトのそれぞれのレベルで使用されている命令コードを指定する必要があります。

子レベルの命令コードと親レベルの命令コードのレベルが等しい場合、動詞に対してグローバル命令コード (親命令コード) が適用されます。そうでない場合は、まず親オブジェクトの親命令コードが適用され、その後それぞれの子オブジェクトの子命令コードが適用されます。

例えば、コンタクトを作成する必要があるときに、Portal_Contact ビジネス・オブジェクトが個別ビジネス・オブジェクトとして送信される場合は、Create 動詞のアプリケーション固有情報に命令コード CUST_SET_NAMEINFO を使用してください。ただし、コンタクトがアカウント・ビジネス・オブジェクトとともに作成される場合は、親命令コード CUST_COMMIT_CUSTOMER を使用してください。この命令コードは、アプリケーション固有情報内に指定する必要があります。上記の機能をサポートするには、Portal_Contact ビジネス・オブジェクトのコピーを 2 つ作成して、2

種類の命令コードに使用します。一方のビジネス・オブジェクトは動詞 ASI を CUST_SET_NAMEINFO に設定し、もう一方のビジネス・オブジェクトは動詞 ASI を CUST_COMMIT_CUSTOMER に設定します。

Portal Infranet ビジネス・オブジェクト定義の完全サンプル

以下は、Portal Infranet アダプターにおける Sample Account ビジネス・オブジェクトのプロパティーおよびアプリケーション固有の情報を記述した構造体です。

```
[BusinessObjectDefinition]
Name = Portal_Account
Version = 1.0.0
AppSpecificInfo = CN=/account
[Attribute]
Name = Poid
Type = String
Cardinality = 1
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = true
AppSpecificInfo = eu
IsRequiredServerBound = false
[End]
[Attribute]
Name = AccountNumber
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=;FN=PIN_FLD_ACCOUNT_NO;Create=true;O=true;DeleteT=;
   Update=falTse;RetrieveT=;Alone=false;CreateT=;Retrieve=false;
   DeleteO=;ParentAtt=;UpdateT=;RetrieveO=Main;Delete=false;CreateO=
IsRequiredServerBound = false
[End]

[Attribute]
Name = AccountObj
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=;FN=PIN_FLD_ACCOUNT_OBJ;Create=true;O=true;DeleteT=;Update=false;
   RetrieveT=;Alone=false;CreateT=;Retrieve=false;DeleteO=;ParentAtt=;
   UpdateT=;RetrieveO=Main;Delete=false;CreateO=
IsRequiredServerBound = false
[End]

[Attribute]
Name = Status
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=;FN=PIN_FLD_STATUS;Create=true;O=true;DeleteT=;Update=true;
   RetrieveT=;Alone=false;CreateT=;DeleteO=;Retrieve=false;ParentAtt=;
   RetrieveO=Main;UpdateT=PIN_FLD_STATUSES;CreateO=;Delete=false
```

```
IsRequiredServerBound = false
[End]

[Attribute]
Name = StatusReason
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=;FN=PIN_FLD_STATUS_FLAGS;Create=true;O=true;DeleteT=;
   Update=true;RetrieveT=;Alone=false;CreateT=;DeleteO=;Retrieve=false;
   ParentAtt=;RetrieveO=Main;UpdateT=PIN_FLD_STATUSES;CreateO=;Delete=false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Portal_Contact
Type = Portal_Contact
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = n
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=Main;FN=PIN_FLD_NAMEINFO;Create=true;O=true;Update=true;
   Alone=false;Retrieve=false;DeleteO=Main;ParentAtt=;RetrieveO=Main;
   CreateO=Main;Delete=false
IsRequiredServerBound = false
[End]

[Attribute]
Name = PlaceHolder
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Portal_BillInfo
Type = Portal_BillInfo
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = n
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=Main;FN=PIN_FLD_BILLINFO;Create=true;O=true;Update=true;
   Alone=false;Retrieve=false;DeleteO=Main;ParentAtt=;RetrieveO=Main;
   CreateO=Main;Delete=false
IsRequiredServerBound = false
[End]

[Attribute]
Name = ProgramName
Type = String
Cardinality = 1
```

```
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=Main;FN=PIN_FLD_PROGRAM_NAME;Create=false;O=true;DeleteT=;
   Update=true;RetrieveT=;Alone=false;CreateT=;DeleteO=Main;Retrieve=false;
   ParentAtt=;PAttName=;RetrieveO=Main;UpdateT=;CreateO=;Delete=true
DefaultValue = CrossWorlds
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
AppSpecificInfo =
   IFM=;OpCode=CUST_COMMIT_CUSTOMER;OFP=;TFlag=true;
   IFP=;IF=NORMAL;Flag=0;OF=NORMAL
[End]

[Verb]
Name = Delete
AppSpecificInfo =
   IFM=;OpCode=CUST_DELETE_ACCT;OFP=;TFlag=false;IFP=;
   IF=NORMAL;Flag=0;OF=NORMAL
[End]

[Verb]
Name = Retrieve
AppSpecificInfo =
   IFM=;OpCode=READ_OBJ;OFP=Portal_BillInfo;TFlag=false;IFP=;
   IF=NORMAL;Flag=0;OF=NORMAL
[End]

[Verb]
Name = Update
AppSpecificInfo =
   IFM=;OpCode=CUST_SET_STATUS;OFP=;TFlag=false;
   IFP=;IF=NORMAL;Flag=0;OF=NORMAL
[End]

[End]
```

第 4 章 PortalODA 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、コネクター用のビジネス・オブジェクト定義を生成するオブジェクト・ディスカバリー・エージェント (ODA) である、PortalODA について説明します。PortalODA は、Portal Infranet API を使用して Portal Infranet 格納可能クラスに関する情報を取得します。その後、この情報を使用して新しいビジネス・オブジェクト定義を作成します。PortalODA により、既存のビジネス・オブジェクト定義をコネクターでサポートされているビジネス・オブジェクト定義に変換することもできます。

以下の項目について説明します。

- 『インストールおよび使用』
- 『PortalODA のインストール』
- 62 ページの『複数のマシン上での PortalODA の実行』
- 63 ページの『Business Object Designer での PortalODA の使用』
- 73 ページの『生成された定義の内容』
- 75 ページの『ビジネス・オブジェクト定義への情報の追加』

インストールおよび使用

このセクションでは、以下について説明します。

- 61 ページの『PortalODA のインストール』
- 62 ページの『PortalODA を使用する前に』
- 62 ページの『PortalODA の始動』
- 62 ページの『複数のマシン上での PortalODA の実行』
- 63 ページの『エラーおよびメッセージ・ファイル名の変更』

PortalODA のインストール

PortalODA をインストールするには、WebSphere Business Integration Adapter (WBIA) のインストーラーを使用します。「システム・インストール・ガイド *(UNIX 版)*」または「システム・インストール・ガイド *(Windows 版)*」に記載されている手順に従ってください。インストールの完了後、使用システム上の製品をインストールしたディレクトリーを調べると、次のファイルがインストールされています。

- `ODA¥Portal¥PortalODA.jar`
- `ODA¥messages¥PortalODAAgent.txt`
- `ODA¥Portal¥start_PortalODA.bat` (Windows のみ)
- `ODA/Portal/start_PortalODA.sh` (UNIX のみ)
- `bin¥CWODAEnv.bat` (Windows のみ)
- `bin/CWODAEnv.sh` (UNIX のみ)

注: 特に指定がない限り、本書ではディレクトリー・パスの記述に円記号 (¥) を使用します。UNIX システムの場合には、円記号 (¥) はスラッシュ (/) に置き換えてください。すべての WBIA 製品のパス名は、使用システムで製品がインストールされたディレクトリーを基準とした相対パス名です。

PortalODA を使用する前に

PortalODA を実行するには、必要な Portal Infranet アプリケーションの .jar ファイルを *%ProductDir%*/connectors/Portal/dependencies ディレクトリーにコピーする必要があります。このディレクトリーには以下のファイルをコピーしてください。

```
pcm.jar
pcmext.jar
```

上記のファイルは %INFRANET%¥jars フォルダーに存在します。

PortalODA のインストール後、ビジネス・オブジェクトを生成または変換するには、以下の手順を行う必要があります。

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

このステップについては、以下のセクションで詳しく説明します。

PortalODA の始動

次のいずれかのスクリプトを使用して PortalODA を始動できます。

UNIX の場合:

```
start_PortalODA.sh
```

Windows の場合:

```
start_PortalODA.bat
```

PortalODA を構成して実行するには、Business Object Designer を使用します。Business Object Designer は、各スクリプト・ファイルまたはバッチ・ファイルの AGENTNAME 変数に指定された名前に基づいて、各 ODA を探し出します。このコネクターのデフォルト ODA の名前は、PortalODA です。

複数のマシン上での PortalODA の実行

ローカル・ホストまたはネットワーク上のリモート・ホストのどちらでも ODA の複数インスタンスを実行することができます。複数のインスタンスを同一マシンで実行する場合、各インスタンスを固有のポートで実行する必要があります。

65 ページの図 16 は、Business Object Designer のウィンドウで、実行する ODA を選択する様子を示したものです。

エラーおよびメッセージ・ファイル名の変更

エラーおよびトレース・メッセージ・ファイル (PortalODAAgent.txt) は、製品ディレクトリーの ¥ODA¥messages¥ に配置されます。このファイルには、次の命名規則が適用されます。

```
AgentNameAgent.txt
```

ODA の名前をスクリプトまたはバッチ・ファイルの AGENTNAME 変数で変更する場合は、この規則を使用して、関連するエラーおよびトレース・メッセージ・ファイルの名前を変更してください。

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する場合は、それぞれについてエラーおよびトレース・メッセージ・ファイルのコピーを作成してください。各ファイルにはこの規則に従って名前を付けてください。例えば、AGENTNAME 変数が PortalODA1 を指定する場合は、関連するメッセージ・ファイルには PortalODA1Agent.txt と命名します。

構成処理では、次のものを指定します。

- PortalODA がエラー情報とトレース情報を書き込むファイルの名前
- トレースのレベル (0 から 5)

表 7 にトレース・レベルの値を示します。

表 7. 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	すべてのメソッド開始/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	作成されたスレッドすべての詳細をトレースします。
5	• すべてのプロパティーの ODA 初期化値を示します。• PortalODA が作成した各スレッドの詳細な状況をトレースします。• ビジネス・オブジェクト定義ダンプをトレースします。

これらの値をどこで構成するかについては、65 ページの『初期プロパティーの構成』を参照してください。

Business Object Designer での PortalODA の使用

このセクションでは、Business Object Designer で PortalODA を使用して、既存のビジネス・オブジェクト定義を新しいビジネス・オブジェクト定義に変換し、新しいビジネス・オブジェクト定義を生成する方法について説明します。これを行うには、Portal Infranet から直接情報を取得します。Business Object Designer の始動については、「*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 の始動後、Business Object Designer を始動させ、ODA を構成し、実行します。Business Object Designer で ODA を使用してビジネス・オブジェクト定義を生成または変換する手順は、6 つのステップから構成されます。Business Object Designer には、これらのステップを順次案内するウィザードが用意されています。

ODA の起動後、このウィザードを起動するには、次の手順を実行します。

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

   Business Object Designer は、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) を表示します。65 ページの図 16 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. 65 ページの『初期プロパティーの構成』
3. 69 ページの『定義の生成』およびオプションで 70 ページの『追加情報の入力』
4. 73 ページの『定義の保管』

ODA の選択

図 16 に、Business Object Designer の 6 段階のウィザードの最初のダイアログ・ボックスを示します。

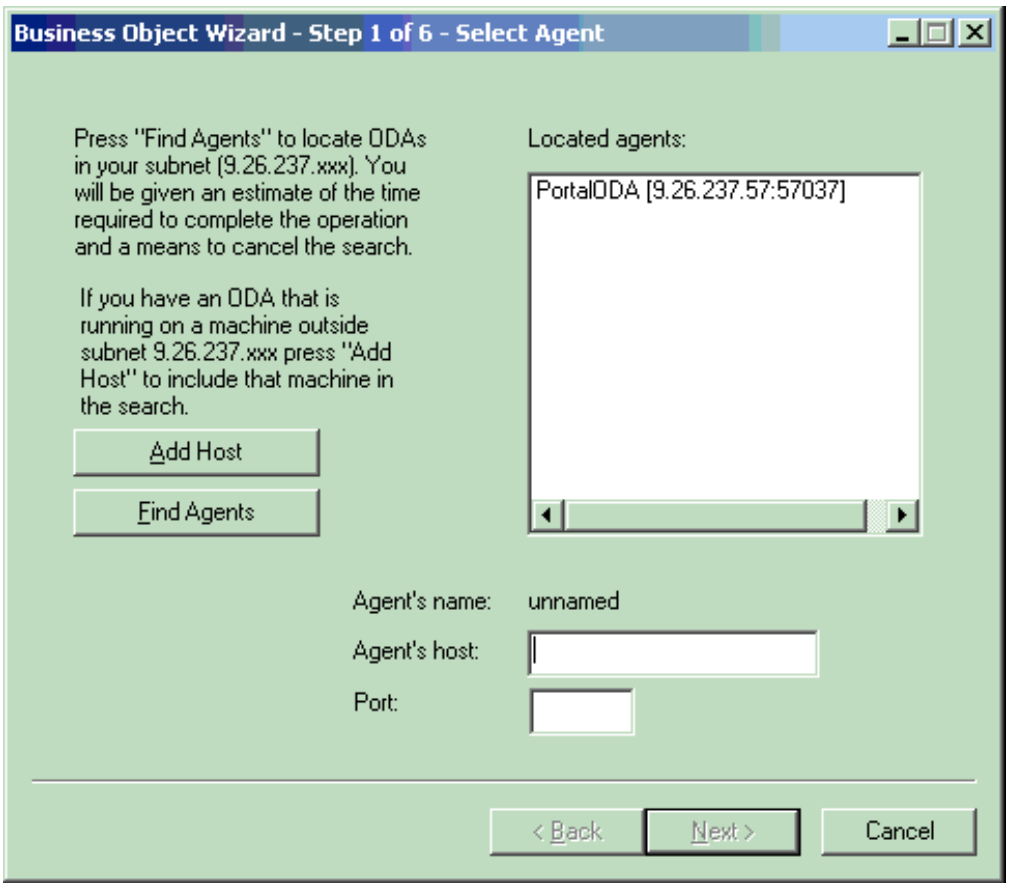


図 16. ODA の選択

ODA を選択するには、次の手順を実行します。

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。

   また、Host 名および Port 番号でエージェントを検索することができます。

   注: Business Object Designer が目的の ODA を見つけられない場合には、ODA の設定をチェックしてください。

2. 表示リストから、目的の ODA を選択します。

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

初期プロパティーの構成

Business Object Designer で最初に PortalODA とやり取りするときに、一連の初期設定プロパティーの入力プロンプトが出されます (図 17 を参照)。これらのプロパティーは、PortalODA を使用するたびに再入力しなくても済むよう、名前つきプロファイルに保存することができます。ODA プロファイルの指定については「*ビジネス・オブジェクト開発ガイド*」を参照してください。

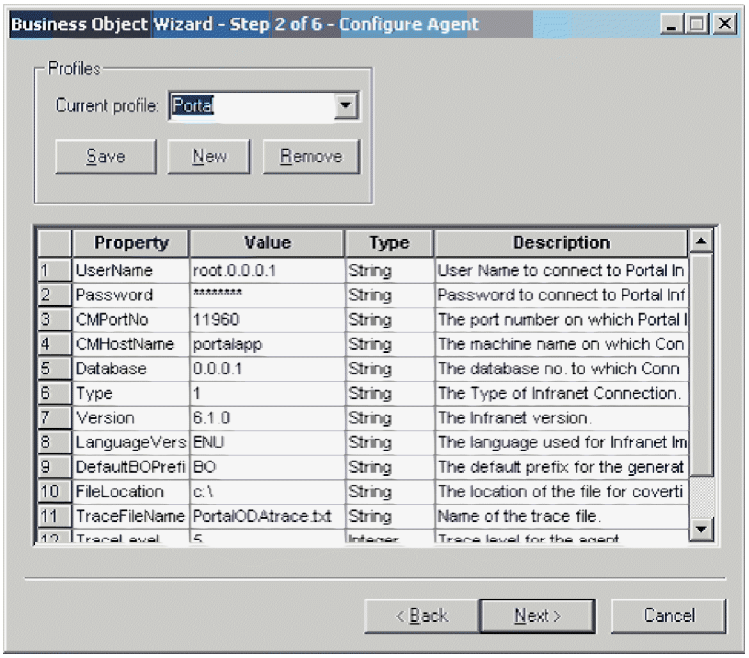


図 17. エージェント初期設定プロパティーの構成

PortalODA プロパティーの構成を表 8 に示します。

重要: 表 8 に示す PortalODA プロパティーはすべて入力する必要があります。

表 8. PortalODA 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	Portal Infranet アプリケーションのログイン名
2	Password	String	Portal Infranet アプリケーションのパスワード
3	CMPortNo	String	接続マネージャーを実行するポート番号
4	CMHostName	String	接続マネージャーを実行するマシンの名前または IP アドレス
5	Database	String	接続マネージャーを接続するデータベース番号
6	Type	String	Portal Infranet 接続タイプ (1 の場合は UserName および Password を検証し、0 の場合は検証しない)
7	Version	String	Portal Infranet のバージョン
8	LanguageVersion	String	例: ENU (英語の場合)
9	DefaultBOPrefix	String	例: Portal_BO
10	FileLocation	String	以前のバージョンのビジネス・オブジェクト定義を持つファイルを含む絶対パス。例えば、Windows においてパスが C:¥PortalBos の場合は、値 C:¥¥Portal¥¥ を入力します。UNIX においてパスが /home/PortalBOs の場合は、値 /home/PortalBos/ を入力します。
11	TraceFileName	String	トレース・ファイルの名前
12	TraceLevel	Integer	ビジネス・オブジェクト名を固有の名前にするために、その先頭に付けられるテキスト。これは、後に、Business Object Designer でビジネス・オブジェクトのプロパティーの入力を要求されたときに、必要に応じて変更することができます。詳細については、70 ページの『追加情報の入力』を参照してください。

表 8. PortalODA 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
13	MessageFile	String	メッセージ・ファイルのパス

ノードの展開、リポジトリー・ファイルおよび格納可能クラスの選択

PortalODA の初期化プロパティーをすべて構成すると、Business Object Designer によって以下の画面が表示されます。

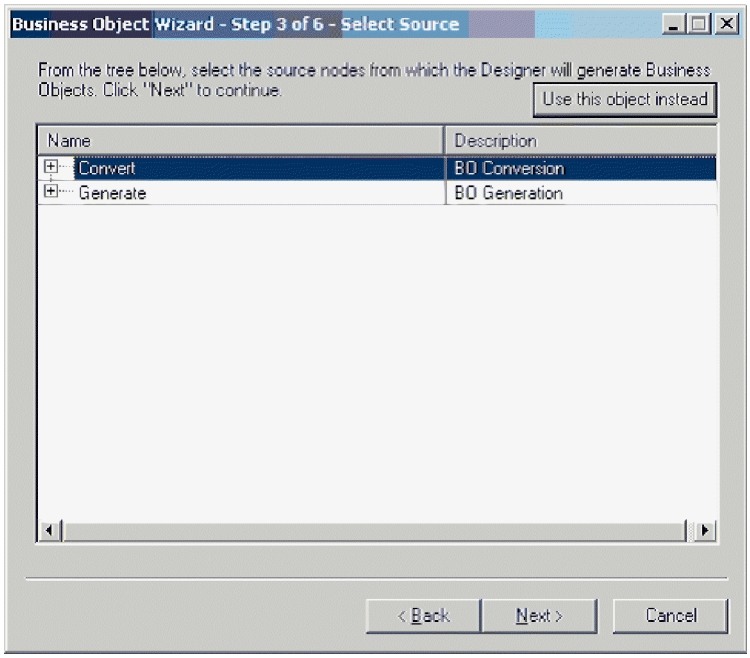


図 18. ビジネス・オブジェクト変換およびビジネス・オブジェクト生成のための 2 つのオプションが表示されたツリー

この画面には、「変換」および「生成」の 2 つの展開可能オプションがあります。古いビジネス・オブジェクト定義を新しいビジネス・オブジェクト定義に変換する必要がある場合は、「変換」を展開します。これにより、変換する必要があるビジネス・オブジェクト定義を含むリポジトリー・ファイルが表示されます。

古いビジネス・オブジェクト定義の変換

古いビジネス・オブジェクト定義のアプリケーション固有情報はコンマで区切られた値であり、新しいビジネス・オブジェクト定義のアプリケーション固有情報は名前と値の組がコンマで区切られた形式になっています。また、古いビジネス・オブジェクト定義はメタビジネス・オブジェクトを使用して特定の命令コードに対するビジネス・オブジェクトの構造を変換しますが、新しいビジネス・オブジェクト定義では、この機能はビジネス・オブジェクトの属性レベルで名前と値の組の形式のアプリケーション固有情報に置き換えられています。

変換するファイルを選択し、「次へ」をクリックします。

注: ファイルを選択すると、そのファイルにあるビジネス・オブジェクト定義がすべて変換されます。変換対象とするビジネス・オブジェクト定義の一部を選択する方法はありません。ただし、ビジネス・オブジェクト定義の一部のみを変換する場合には、ビジネス・オブジェクト定義の一部を含む新規ファイルを作成してからその新規ファイルを変換することができます。

新規ビジネス・オブジェクトの生成

Portal Infranet から情報を取得することによって新規ビジネス・オブジェクト定義を生成する必要がある場合は、「生成」を展開します。これにより、Portal Infranet からの格納可能クラス名がすべて取得され、ツリーに表示されます。

ツリーでノードとして表示されている格納可能クラス名は展開可能です (図 19 を参照してください)。生成されたビジネス・オブジェクトをコネクターで使用するには、そのビジネス・オブジェクトの一部のプロパティーを個別に設定する必要があります。ビジネス・オブジェクトのキー・フィールドは、必ず WebSphere Business Integration システムのビジネス・オブジェクトでキー・フィールドとしてマークする必要があります。動詞ごとに使用する命令コードによっては、属性レベルのアプリケーション固有情報を設定する必要があります。例えば、属性が Create 動詞の命令コードの一部である場合は、プロパティー「Create」の値を親フィールドの名前に設定する必要があります。属性のアプリケーション固有情報の各種のプロパティーの詳細については、 44 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。

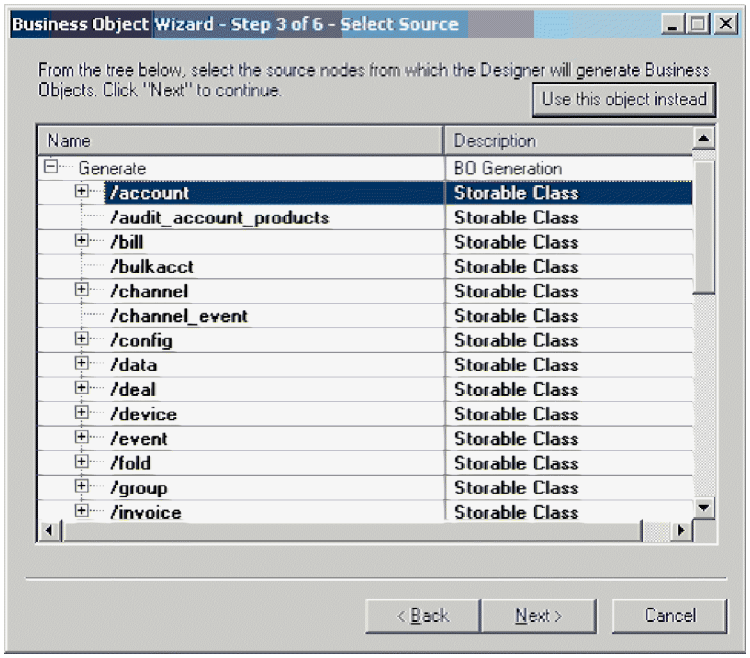


図 19. 格納可能クラスを表示する画面

この画面では、生成する格納可能クラスをリストから選択できます。クラス名の前の「+」記号は、そのクラスに子オブジェクトがあることを示します。複数のクラスの生成を選択できます。

注: 生成するクラスとして子オブジェクトを持つクラスを選択した場合、デフォルトでは子オブジェクトは選択されません。子オブジェクトも生成させる場合

は、明示的に子オブジェクトを選択する必要があります。これを行うには、シフト・キーを押しながら子オブジェクトを選択します。

リポジトリー・ファイルおよび格納可能クラスの選択の確認

生成するビジネス・オブジェクト定義に関連付けるリポジトリー・ファイルまたは格納可能クラスをすべて識別すると、Business Object Designer は以下の確認画面を表示します (図 20 を参照してください)。

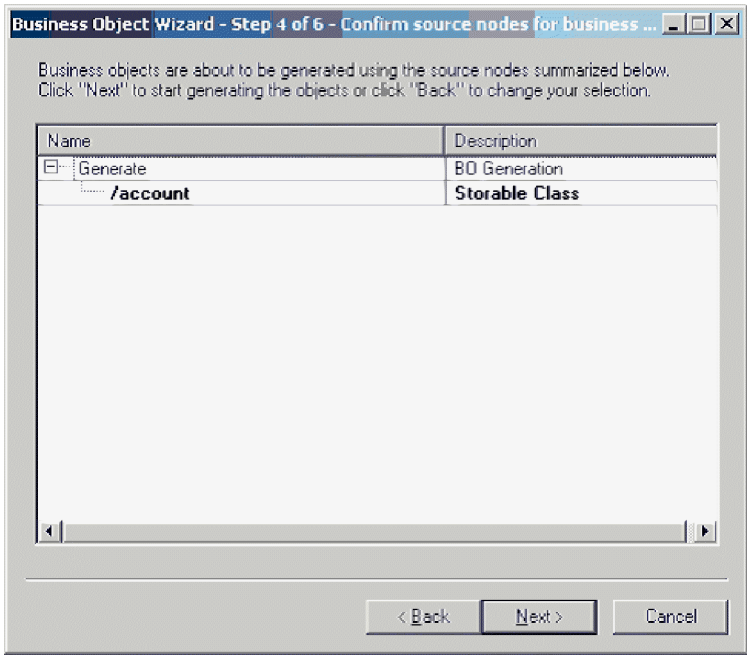


図 20. 選択の確認

このウィンドウには、次のオプションが用意されています。

- 選択内容を確認するには、「次へ」をクリックします。
- 選択内容が正しくない場合は、「戻る」をクリックして直前のウィンドウに戻り、必要な変更を行います。選択内容が訂正されたら、「次へ」をクリックします。

定義の生成

選択したデータベース・オブジェクトを確認すると、次のダイアログ・ボックスが開き、Business Object Designer によって定義が生成されていることを通知します。

図 21 にこのダイアログ・ボックスを示します。

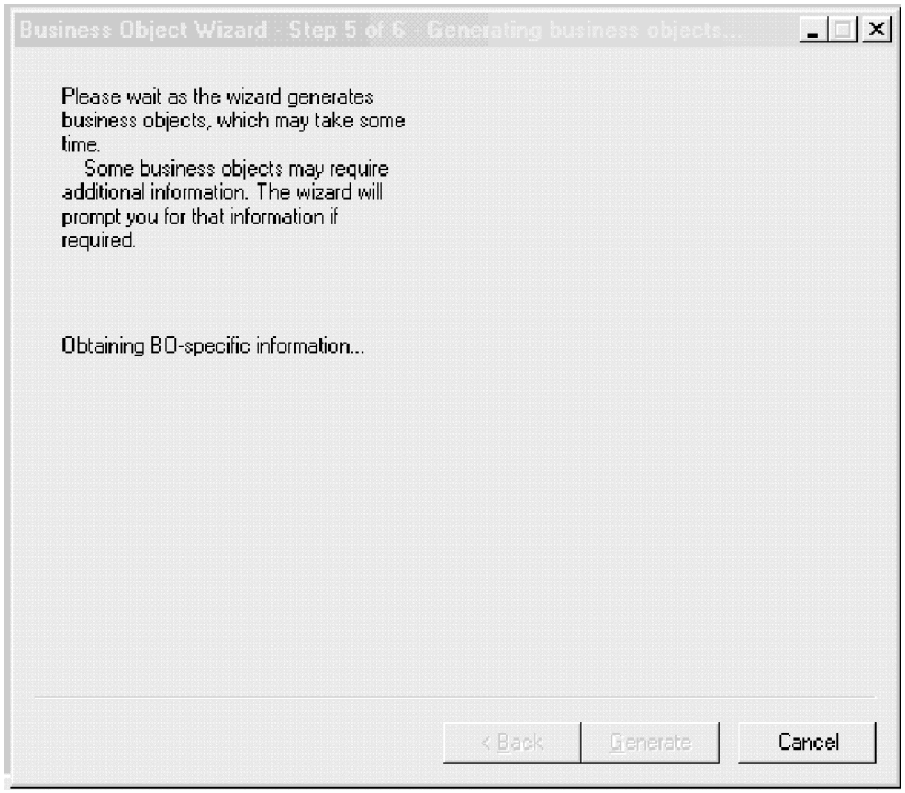


図 21. 定義の生成

追加情報の入力

PortalODA に追加情報が必要な場合、Business Object Designer では、「BO プロパティー」ウィンドウを表示し、ユーザーにその情報の入力を要求します。これは、ビジネス・オブジェクト生成の場合にのみ行います。図 22 にこのウィンドウを示します。

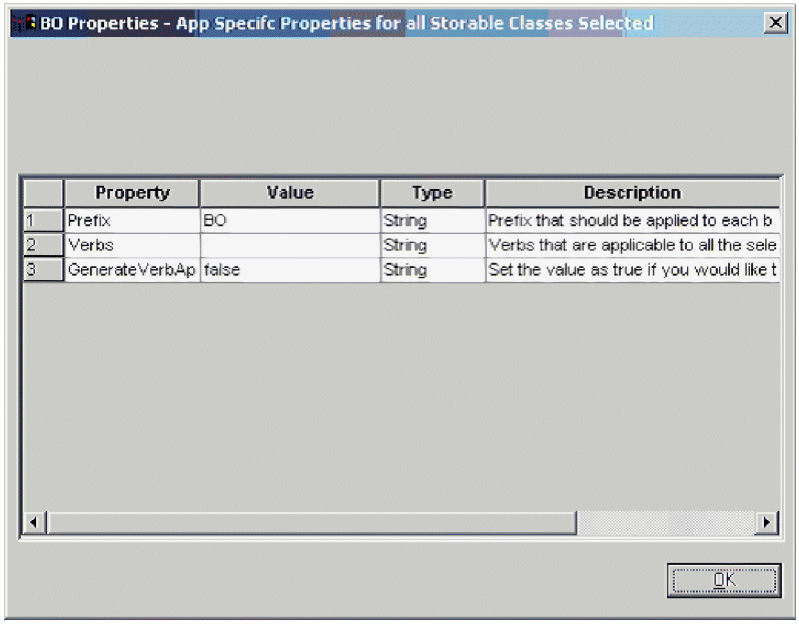


図 22. 格納可能クラスに関するアプリケーション固有プロパティー

「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。

- *Prefix:* ビジネス・オブジェクト名を固有の名前にするために、その先頭に付けられるテキスト。「エージェントの構成」ウィンドウ (図 17) で *DefaultBOPrefix* プロパティーの値として入力した値を使用しても不都合がない場合は、ここで値を変更する必要はありません。
- *Verbs:*「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になります。

  注: 「BO プロパティー」ダイアログ・ボックスに複数の値を含むフィールドがある場合、そのフィールドは、このダイアログ・ボックスが開いた時点では、空であるかのように表示されます。フィールド内をクリックすると、含まれる値を示すドロップダウン・リストが表示されます。
- *GenerateVerbApp:* 動詞レベルでアプリケーション固有情報を編集できるフラグ。

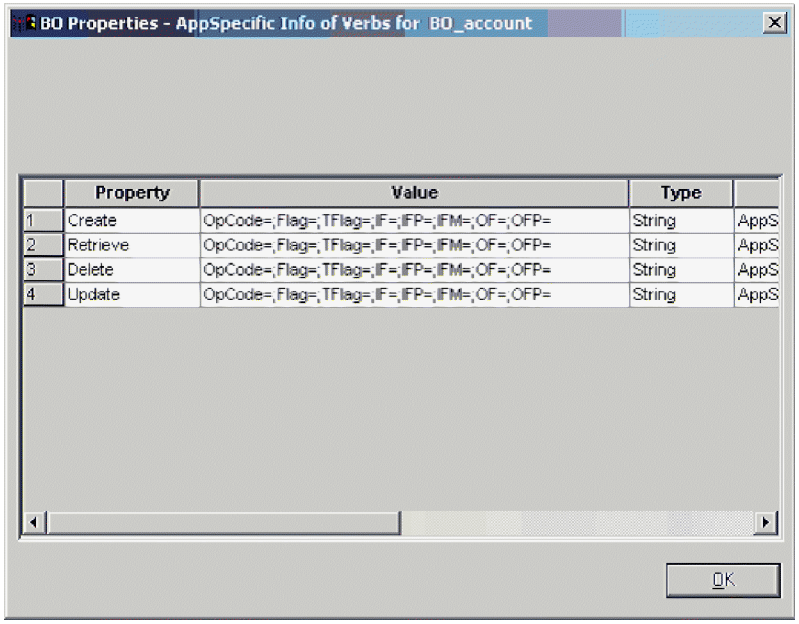


図 23. 動詞に関するアプリケーション固有情報

動詞レベルのアプリケーション固有情報のフォーマットは以下のとおりです。

```
OpCode=;Flag=;TFlag=;IF=;IFP=;IFM=;OF=OFP= describes
```

表 9 に、動詞レベルのアプリケーション固有情報のそれぞれの名前を示します。

表 9. 動詞に関するアプリケーション固有情報

名前	説明
Opcode	この動詞の場合に実行する命令コードの名前
Flag	命令コードで使用するフラグ値
TFlag	TFlag は、命令コードが固有のトランザクションを維持するかどうかに応じて true または false をとります
IF	入力フィールド・リスト (IF) は、命令コードの入力フィールド・リストの作成に使用するビジネス・オブジェクトの名前です
IFP	入力フィールド・リスト・パラメーター (IFP) は、入力フィールド・リストの作成に使用できるオプショナル・パラメーターの名前です
IFM	入力フィールド・リスト・モード (IFM) は、行うフィールド・リスト変換の種類を定義する値です
OF	出力フィールド・リスト (OF) は、命令コード実行の戻りフィールド・リストをビジネス・オブジェクトに変換する方法を制御するパラメーターです
OFP	出力フィールド・リスト・モード (OFM) は、命令コードの出力フィールド・リストから実行するビジネス・オブジェクト更新の種類を定義する値です

定義の保管

「BO プロパティー」ダイアログ・ボックスに必要な情報をすべて入力して「OK」をクリックすると、Business Object Designer には、ウィザードの最後のウィンドウが表示されます。ここで、定義をサーバーまたはファイルに保管することができます。あるいは、Business Object Designer 内で定義を開き、編集することができます。これらの詳細、およびさらに変更を行う方法については「*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 24 にこのダイアログ・ボックスを示します。

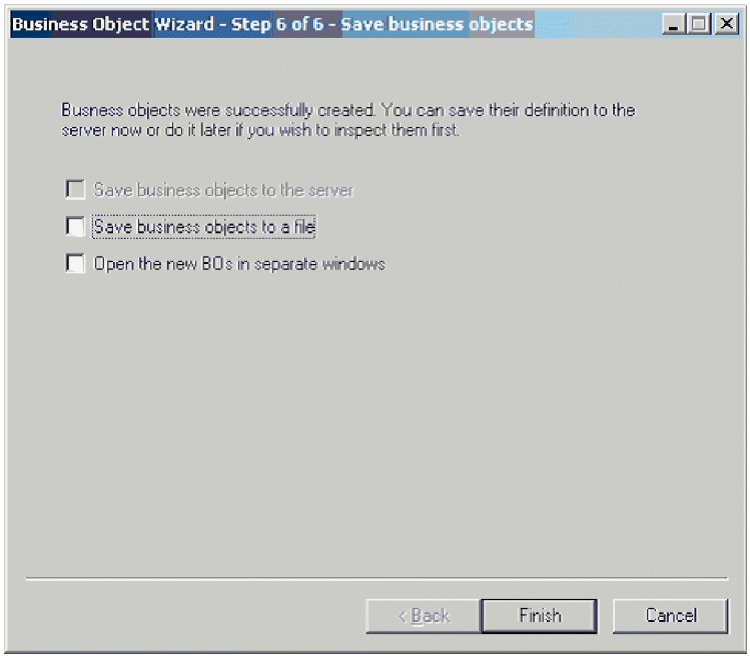


図 24. ビジネス・オブジェクト定義の保管

生成された定義の内容

PortalODA によって生成されたビジネス・オブジェクト定義には、次のものが含まれます。

- 指定されたデータベース表およびビューの列に対応する属性 (1 列につき 1 属性)
- 「BO プロパティー」ウィンドウで指定された動詞 (図 23)
- アプリケーション固有情報
  - ビジネス・オブジェクト・レベルの情報
  - 属性ごとの情報
  - 動詞ごとの情報

Portal Infranet から情報を取得することによってビジネス・オブジェクトを生成する場合は、生成されるアプリケーション固有情報は単純属性のもののみです。この規則の例外は、コンテナー属性が複数値リンクである場合です。他の場合は、いずれ

もユーザーがアプリケーション固有情報を入力する必要があります (35 ページの『第 3 章 ビジネス・オブジェクトの理解』を参照してください)。

このセクションで説明する内容は次のとおりです。

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

ビジネス・オブジェクト・レベルのプロパティー

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

- ビジネス・オブジェクト名
- バージョン (デフォルトで 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・レベルのアプリケーション固有情報には、対応する Portal Infranet ビジネス・コンポーネントの名前が含まれます。

属性プロパティー

このセクションでは、PortalODA が属性ごとに生成するプロパティーについて説明します。

重要: 以下のセクションで説明するユーザーによる編集は、ビジネス・オブジェクト変換ではなく、ビジネス・オブジェクト生成のみを対象としています。

Name プロパティー

PortalODA は、Portal Infranet ビジネス・コンポーネントの対応する属性から属性の名前の値を取得します。

Data type プロパティー

このタイプの属性を設定すると、PortalODA は Portal Infranet ビジネス・コンポーネントの属性のデータ型を変換し、対応するデータ型に変換します (表 10 を参照してください)。ビジネス・オブジェクト変換は既存のビジネス・オブジェクトを対象とするため、この処理はビジネス・オブジェクト生成の場合にのみ行われます。

表 10. データ型の対応

アプリケーション	WebSphere Business Integration システム	長さ
PIN_FLDT_INT	Integer	
PIN_FLDT_ENUM	Integer	
PIN_FLDT_STR	String	Portal Infranet の対応する属性の長さ
PIN_FLDT_BUF	String	Portal Infranet の対応する属性の長さ
PIN_FLDT_POID	String	Portal Infranet の対応する属性の長さ
PIN_FLDT_TSTAMP	Date	
PIN_FLDT_ARRAY	Object	
PIN_FLDT_SUBSTRUCT	Object	
PIN_FLDT_BINSTR	String	Portal Infranet の対応する属性の長さ

表 10. データ型の対応 (続き)

PIN_FLDT_DECIMAL	Float	

注: 属性のデータ型が、表 10 に含まれるデータ型以外のものである場合、PortalODA はその列をスキップし、その列を処理できないというメッセージを表示します。

Cardinality プロパティー

PortalODA 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、コンテナー属性を n に設定します。コンテナー属性のカーディナリティーを変更する必要がある場合は、ユーザー側で変更する必要があります。

MaxLength プロパティー

PortalODA は Portal Infranet から属性の長さを取得します。

IsKey プロパティー

PortalODA は属性にキー・フィールドのマークを付けません。ビジネス・オブジェクトが生成された後、手動でキー・フィールドにマークを付ける必要があります。

IsRequired プロパティー

テーブルまたはビューに非ヌルに指定されているフィールドがある場合、PortalODA は、そのフィールドを必須属性としてマークします。しかし、Portal Infranet アプリケーションはレコード作成中に固有の ID 値を生成するため、PortalODA はキー・フィールドを必須としてマークしません。

AppSpecificInfo プロパティー

ユーザーは、コンテナー属性が生成されていない場合はこのプロパティーを編集し、コンテナー属性が生成された場合は正確かどうかを確認する必要があります。

動詞

PortalODA は、「BO プロパティー」ウィンドウ (72 ページの図 23) で指定された動詞を生成します。各動詞の AppSpecificInfo プロパティーを作成しますが、設定は行いません。

ビジネス・オブジェクト定義への情報の追加

Portal Infranet 格納可能クラスにはビジネス・オブジェクトが必要とする情報の一部が存在しないことがあるため、特に新規ビジネス・オブジェクトを生成した場合には、PortalODA が作成するビジネス・オブジェクト定義に情報を追加する必要があることがあります。

ビジネス・オブジェクト定義を検証したり変更した定義をリポジトリーに再ロードしたりするには、Business Object Designer を使用します。

注: あるいは、ICS が統合ブローカーである場合は、repos_copy コマンドを使用して定義をリポジトリーにロードできます。WebSphere MQ Integrator Broker が統合ブローカーである場合は、システム・コマンドを使用してファイルをリポジトリー・ディレクトリーにコピーできます。

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

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

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

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

新規プロパティーと削除されたプロパティー

以下の標準プロパティーは、本リリースで追加されました。

新規プロパティー

- XMLNameSpaceFormat

削除されたプロパティー

- RestartCount
- RHF2MessageDomain

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 の使用

Connector Configurator からコネクター・プロパティーを構成します。Connector Configurator には、System Manager からアクセスします。Connector Configurator の使用法の詳細については、付録の『Connector Configurator』を参照してください。

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

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 (WebSphere InterChange Server が統合ブローカーである場合のみ)
3. ローカル構成ファイル
4. コマンド行

コネクターは、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。コネクターが System Manager から独立してスタンドアロン・モードで稼働している場合 (例えば、いずれかの WebSphere Message Brokers と連携している場合) は、構成ファイルでのみプロパティーを変更できます。この場合、動的更新は実行できません。
- 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 (ICS のみ)
  アプリケーション固有のコンポーネントを停止して再始動しなければ、変更が有効になりません。

特定のプロパティーの更新方法を確認するには、「Connector Configurator」ウィンドウ内の**「更新メソッド」**列を参照するか、次に示す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 11 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 11. 標準構成プロパティーの要約

プロパティー名	可能な値	デフォルト値	更新メソッド	注
AdminInQueue	有効な JMS キュー名	*CONNECTORNAME* /ADMININQUEUE	コンポーネント再始動	Delivery Transport は JMS
AdminOutQueue	有効な JMS キュー名	*CONNECTORNAME*/ADMINOUTQUEUE	コンポーネント再始動	Delivery Transport は JMS
AgentConnections	1 から 4	1	コンポーネント再始動	Delivery Transport は MQ および IDL: Repository Directory は <REMOTE>
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、WMQI、WAS			
CharacterEncoding	ascii7、ascii8、SJIS、Cp949、GBK、Big5、Cp297、Cp273、Cp280、Cp284、Cp037、Cp437 **注:** これは、サポートされる値の一部です。	ascii7	コンポーネント再始動	
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント再始動	Repository Directory は <REMOTE>
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	MQ、IDL、または JMS	JMS	コンポーネント再始動	Repository Directory がローカルの場合は、値は JMS のみ

表 11. 標準構成プロパティーの要約 (続き)

プロパティー名	可能な値	デフォルト値	更新メソッド	注
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
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory または CxCommon.Messaging.jms.SonicMQFactory または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	FactoryClassName が IBM の場合は crossworlds.queue.manager を使用。FactoryClassName が Sonic の場合 localhost:2506 を使用。	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	Repository Directory は <REMOTE>
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>
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならない
Locale	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR **注:** これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	True または False	False	コンポーネント再始動	Repository Directory は <REMOTE> でなければならない

表 11. 標準構成プロパティーの要約 (続き)

プロパティー名	可能な値	デフォルト値	更新メソッド	注
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならない
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> でなければならない
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければならない
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければならない
PollEndTime	HH:MM	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒) no (ポーリングを使用不可にする) key (コネクターのコマンド・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	ICS の場合は <REMOTE> に設定する。WebSphere MQ Message Brokers および WAS の場合: C:¥crossworlds¥repository に設定する

表 11. 標準構成プロパティーの要約 (続き)

プロパティー名	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 が JMS の場合: Repository Directory が <REMOTE> の場合のみ必要
RestartRetryCount	0 から 99	3	動的	
RestartRetryInterval	適切な正数 (単位: 分): 1 から 2147483547	1	動的	
SourceQueue	有効な WebSphere MQ 名	*CONNECTORNAME*/SOURCEQUEUE	エージェント再始動	Delivery Transport が JMS であり、かつ Container Managed Events が指定されている場合のみ
SynchronousRequestQueue		*CONNECTORNAME*/ SYNCHRONOUSREQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
SynchronousRequestTimeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	Delivery Transport は JMS
SynchronousResponseQueue		*CONNECTORNAME*/ SYNCHRONOUSRESPONSEQUEUE	コンポーネント再始動	Delivery Transport は JMS
WireFormat	CwXML、CwBO	CwXML	エージェント再始動	Repository Directory が <REMOTE> でない場合は CwXML。Repository Directory が <REMOTE> であれば CwBO
WsifSynchronousRequest Timeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	WAS のみ
XMLNameSpaceFormat	short、long	short	エージェント再始動	WebSphere MQ Message Brokers および WAS のみ

標準構成プロパティー

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

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

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・タイプを指定します。オプションは ICS、WebSphere Message Brokers (WMQI、WMQIB または WBIMB) または WAS です。

CharacterEncoding

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

注: Java ベースのコネクターでは、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、Connector Configurator に関する付録を参照してください。

ConcurrentEventTriggeredFlows

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

コネクタ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。

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 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。

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

ControllerStoreAndForwardMode

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

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

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

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

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

デフォルト値は true です。

ControllerTraceLevel

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

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

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

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

- ICS がブローカー・タイプの場合は、DeliveryTransport プロパティーの指定可能な値は MQ、IDL、または JMS であり、デフォルトは IDL になります。

- RepositoryDirectory がローカル・ディレクトリーの場合は、指定可能な値は JMS のみです。

DeliveryTransport プロパティーに指定されている値が、MQ または 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。

WebSphere MQ および IDL

イベントのデリバリー・トランスポートには、IDL ではなく WebSphere MQ を使用してください (1 種類の製品だけを使用する必要がある場合を除きます)。
WebSphere MQ が IDL よりも優れている点は以下のとおりです。

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

JMS

Java Messaging Service (JMS) を使用しての、コネクターとクライアント・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した場合は、
jms.MessageBrokerName、jms.FactoryClassName、jms.Password、jms.UserName などの追加の JMS プロパティーが Connector Configurator 内に表示されます。このうち最初の 2 つは、このトランスポートの必須プロパティーです。

重要: 以下の環境では、コネクターに 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もあります。

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

この環境では、WebSphere MQ クライアント内でメモリーが使用されるため、(サーバー側の) コネクター・コントローラーと (クライアント側の) コネクターの両方を

始動するのは困難な場合があります。ご使用のシステムのプロセス・ヒープ・サイズが 768M 未満である場合には、次のように設定することをお勧めします。

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

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

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

  この行は、ヒープ・メモリーの使用量を最大 768 MB (3 セグメント * 256 MB) に制限します。プロセス・メモリーがこの制限値を超えると、ページ・スワッピングが発生し、システムのパフォーマンスに悪影響を与える場合があります。
- IPCCBaseAddress プロパティーの値を 11 または 12 に設定する。このプロパティーの詳細については、「*システム・インストール・ガイド (UNIX 版)*」を参照してください。

DuplicateEventElimination

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

このプロパティーは、false に設定することもできます。

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

FaultQueue

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

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

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

デフォルト値は 128M です。

JvmMaxNativeStackSize

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

デフォルト値は 128K です。

JvmMinHeapSize

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

デフォルト値は 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 (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 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。

ListenerConcurrency

このプロパティーは、統合ブローカーとして ICS を使用する場合の MQ Listener でのマルチスレッド化をサポートしています。このプロパティーにより、データベースへの複数イベントの書き込み操作をバッチ処理できるので、システム・パフォーマンスが向上します。デフォルト値は 1 です。

このプロパティーは、MQ トランスポートを使用するコネクターにのみ適用されます。DeliveryTransport プロパティーには MQ を設定してください。

Locale

言語コード、国または地域、および、希望する場合には、関連した文字コード・セットを指定します。このプロパティーの値は、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、次の書式で指定します。

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、Connector Configurator 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

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

統合ブローカーのログ宛先にエラーを記録するかどうかを指定します。ブローカーのログ宛先にログを記録すると、電子メール通知もオンになります。これにより、エラーまたは致命的エラーが発生すると、InterchangeSystem.cfg ファイルに指定された MESSAGE_RECIPIENT に対する電子メール・メッセージが生成されます。

例えば、LogAtInterChangeEnd を true に設定した場合にコネクタ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が使用し、RepositoryDirectory プロパティーの値が <REMOTE> の場合にのみ適用されます。

値は 1 から 2147483647 の間の正整数です。デフォルト値は 2147483647 です。

MessageFileName

コネクター・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

RepositoryDirectory が <REMOTE> の場合のみ有効です。

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)*」または「*システム・インストール・ガイド (UNIX 版)*」を参照してください。

デフォルト値は false です。

OADMaxNumRetry

RepositoryDirectory が <REMOTE> の場合のみ有効です。

異常シャットダウンの後で MQ により起動される OAD がコネクタ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

RepositoryDirectory が <REMOTE> の場合のみ有効です。

MQ により起動される OAD の再試行時間間隔の分数を指定します。コネクター・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、該当するアダプター・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

統合ブローカーが、ビジネス・オブジェクトをコネクターに送信するときに使用されるキューです。

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

コネクターが 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。

統合ブローカーが ICS の場合はこの値を <REMOTE> に設定する必要があります。これは、コネクターが InterChange Server リポジトリーからこの情報を取得するためです。

統合ブローカーが WebSphere Message Broker または WAS の場合は、この値を <*local directory*> に設定する必要があります。

ResponseQueue

DeliveryTransport が JMS の場合のみ適用可能で、RepositoryDirectory が <REMOTE> の場合のみ必須です。

JMS 応答キューを指定します。JMS 応答キューは、応答メッセージをコネクター・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。統合ブローカーが ICS の場合、サーバーは要求を送信し、JMS 応答キューの応答メッセージを待ちます。

RestartRetryCount

コネクタ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、84 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配信します。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・フレームワークに配信します。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。

- RepositoryDirectory がローカル・ディレクトリーの場合は、設定は CwXML になります。
- RepositoryDirectory の値が <REMOTE> の場合には、設定値は CwBO です。

WsifSynchronousRequest Timeout

WAS 統合ブローカーでのみ使用されます。

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。

XMLNameSpaceFormat

WebSphere Message Brokers および WAS 統合ブローカーでのみ使用されます。

ビジネス・オブジェクト定義の XML 形式でネーム・スペースを short と long のどちらにするかをユーザーが指定できるようにするための、強力なプロパティーです。

デフォルト値は `short` です。

付録 B. Connector Configurator

この付録では、Connector Configurator 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。

Connector Configurator 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
- 構成ファイルを作成する
- 構成ファイル内のプロパティーを設定する

注:

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

この付録では、次のトピックについて説明します。

Connector Configurator の概要

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

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

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

- コネクタ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、ICS の場合はコラボレーションとともに使用するマップを

指定し、必要に応じてメッセージング、ロギング、トレース、およびデータ・ハンドラー・パラメーターを指定する必要があります。

Connector Configurator の実行モードと使用する構成ファイルのタイプは、実行する統合ブローカーによって異なります。例えば、使用している統合ブローカーが WMQI の場合、Connector Configurator を System Manager から実行するのではなく、直接実行します (96 ページの『スタンドアロン・モードでの Configurator の実行』を参照)。

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により使用されるので、標準プロパティーを新規に定義する必要はありません。ファイルを作成すると、Connector Configurator により標準プロパティーがこの構成ファイルに挿入されます。ただし、Connector Configurator で各標準プロパティ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 の「標準のプロパティー」ウィンドウには、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。システム内で既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、97 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

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

Connector Configurator の始動

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

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

スタンドアロン・モードでの Configurator の実行

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

これを行うには、以下のステップを実行します。

- **「スタート」>「プログラム」**から、**「IBM WebSphere InterChange Server」>「IBM WebSphere Business Integration Toolset」>「開発」>「Connector Configurator」**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。

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

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (102 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

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。Connector Configurator が開き、この構成ファイルの統合ブローカー・タイプおよびファイル名が上部に表示されます。
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、97 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・ボックスを開かない限り、そのプロパティーの変更内容は受け入れられませんので、注意してください。
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。Connector Configurator により、XML 文書として入力した情報が、Connector Configurator がインストールされている ¥bin ディレクトリーの ¥data¥app の下に保管されます。

新しい構成ファイルを作成

構成ファイルを新規に作成するには、最初に統合ブローカーを選択します。選択したブローカーによって、構成ファイルに記述されるプロパティーが決まります。

ブローカーを選択するには、以下のステップを実行します。

- Connector Configurator のホーム・メニューで、**「ファイル」>「新規」>「コネクター構成」**をクリックします。**「新規コネクター」**ダイアログ・ボックスが表示されます。
- **「Integration Broker」**フィールドで、ICS 接続、WebSphere Message Brokers 接続、WAS 接続のいずれかを選択します。
- この章で後述する説明に従って**「新規コネクター」**ウィンドウの残りのフィールドに入力します。

また、以下の作業も実行できます。

- 「System Manager」ウィンドウで**「コネクター」**フォルダーを右クリックし、**「新規コネクターの作成」**を選択します。Connector Configurator 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。

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 では、入力された名前のスペルはチェックされません。名前が正しいことを確認してください。
   - **システム接続**

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

- **「コネクタ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 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」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

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)。
- ICS リポジトリー・ファイル。
  コネクターの以前の ICS インプリメンテーションで使用した定義は、そのコネクターの構成で使用されたリポジトリー・ファイルで使用可能になります。そのようなファイルの拡張子は、通常 `.in` または `.out` です。
- コネクターの以前の構成ファイル。
  これらのファイルの拡張子は、通常 *.cfg です。

これらのいずれのファイル・ソースにも、コネクタ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 でそのファイルを開き、構成を修正し、そのファイルを再度保管する必要があります。

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

1. Connector Configurator 内で、**「ファイル」>「開く」>「ファイルから」**をクリックします。
2. **「ファイル・コネクターを開く」**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)
   - ICS リポジトリー (*.in、*.out)

     ICS 環境でのコネクターの構成にリポジトリー・ファイルが使用された場合には、このオプションを選択します。リポジトリー・ファイルに複数のコネクタ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 を始動します。
3. **「ファイル」>「開く」>「プロジェクトから」**をクリックします。

構成ファイルの完成

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

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

1. **「標準のプロパティー」**タブで、BrokerType プロパティーの値フィールドを選択します。ドロップダウン・メニューで、値 ICS、WMQI、または WAS を選択します。
2. 選択したブローカーに関連付けられているプロパティーが「標準のプロパティー」タブに表示されます。ここでファイルを保管するか、または 105 ページの

『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。

3. 構成が完了したら、**「ファイル」>「保管」>「プロジェクトに」**を選択するか、または**「ファイル」>「保管」>「ファイルに」**を選択します。

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

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

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

構成ファイル・プロパティーの設定

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

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

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

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

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

- 関連付けられたマップ
- リソース
- メッセージング (該当する場合)

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

標準プロパティ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」フィールドは構成できます。
- **「更新メソッド」**フィールドは通知用であり、構成できません。このフィールドは、値が変更されたプロパティーをアクティブにするために必要なアクションを示します。

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 を終了するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「いいえ」**をクリックします。
   - Connector Configurator 内の他のカテゴリーの値を入力するには、そのカテゴリーのタブを選択します。**「標準のプロパティー」**(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. 104 ページの『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『コネクターの*標準構成プロパティー*』の 78 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。

サポートされるビジネス・オブジェクト定義の指定

コネクターで使用するビジネス・オブジェクトを指定するには、Connector Configurator の**「サポートされているビジネス・オブジェクト」**タブを使用します。

汎用ビジネス・オブジェクトと、アプリケーション固有のビジネス・オブジェクトの両方を指定する必要があり、またそれらのビジネス・オブジェクト間のマップの関連を指定することが必要です。

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

ご使用のブローカーが ICS の場合

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

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

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

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

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

サポートされるリストからビジネス・オブジェクトを削除すると、コネクター定義が変更され、削除されたビジネス・オブジェクトはコネクターのこのインプリメンテーション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」ウィンドウでは「エージェント・サポート」の選択の妥当性は検査されません。

最大トランザクション・レベル: コネクタ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。

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

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

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

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

ご使用のブローカーが WAS の場合

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

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

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

関係付けられたマップ (ICS のみ)

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

ビジネス・オブジェクトのリストには、エージェントでサポートされるアプリケーション固有のビジネス・オブジェクトと、コントローラーがサブスクライブ・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」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、関連付けられたマップを明示的にバインドすることが必要になります。

  明示的バインディングが必要なのは、特定のサポートされるビジネス・オブジェクトに複数のマップが存在する場合のみです。ICS は、ブート時、各コネクターでサポートされるそれぞれのビジネス・オブジェクトにマップを自動的にバインドしようとします。複数のマップでその入力データとして同一のビジネス・オブジェクトが使用されている場合、サーバーは、他のマップのスーパーセットである 1 つのマップを見付けて、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」ウィンドウの**「ファイル」**メニューで、**「プロジェクトに保管」**をクリックします。
4. プロジェクトを ICS に配置します。
5. 変更を有効にするため、サーバーをリブートします。

リソース (ICS)

「リソース」タブでは、コネクター・エージェントが、コネクター・エージェント並列処理を使用して同時に複数のプロセスを処理するかどうか、またどの程度処理するかを決定する値を設定できます。
すべてのコネクターがこの機能をサポートしているわけではありません。複数のプロセスを使用するよりも複数のスレッドを使用する方が通常は効率的であるため、Java でマルチスレッドとして設計されたコネクター・エージェントを実行している場合、この機能を使用することはお勧めできません。

メッセージング (ICS)

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

トレース/ログ・ファイル値の設定

コネクタ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 は、そのファイルのログおよびトレースの値をデフォルト値として使用します。Connector Configurator 内でこれらの値を変更できます。

ログとトレースの値を変更するには、以下の手順を実行します。

1. **「トレース/ログ・ファイル」**タブをクリックします。
2. ログとトレースのどちらでも、以下のいずれかまたは両方へのメッセージの書き込みを選択できます。
   - コンソールに (STDOUT):
     ログ・メッセージまたはトレース・メッセージを STDOUT ディスプレイに書き込みます。

     注: STDOUT オプションは、Windows プラットフォームで実行しているコネクターの**「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。すべてのアダプターでデータ・ハンドラーを使用できるわけではありません。

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

構成ファイルの保管

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

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

- System Manager から、統合コンポーネント・ライブラリーに *.con 拡張子付きファイルとして保管します。
- System Manager から、指定したディレクトリーに *.con 拡張子付きファイルとして保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに *.cfg 拡張子付きファイルとして保管します。

System Manager でのプロジェクトの使用法、および配置の詳細については、以下のインプリメンテーション・ガイドを参照してください。

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

構成ファイルの変更

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

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

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

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

構成の完了

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

これを行うには、コネクタ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 の使用

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

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

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

CharacterEncoding および Locale 標準構成プロパティーのドロップ・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>
```

付録 C. 特記事項

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、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。

汎用プログラミング・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
Domino
Lotus
Lotus Notes
MQIntegrator
MQSeries
Tivoli
WebSphere

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

MMX、Pentium および ProShare は、Intel Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

WebSphere Business Integration Adapter Framework V2.4.0

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