IBM WebSphere Business Integration Server Express and Express Plus

Adapter for JD Edwards OneWorld ユーザーズ・ガイド

バージョン 2.0.x

IBM WebSphere Business Integration Server Express and Express Plus

Adapter for JD Edwards OneWorld ユーザーズ・ガイド

バージョン 2.0.x

お願い

本書および本書で紹介する製品をご使用になる前に、 105 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere Business Integration Server Express and Express Plus Adapter for JD Edwards One World バージョン 2.0.x に適用されます。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

http://www.ibm.com/jp/manuals/main/mail.html

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

http://www.ibm.com/jp/manuals/ の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

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

原　典： IBM WebSphere Business Integration Server Express
and Express Plus
Adapter for JD Edwards OneWorld User Guide
Version 2.0.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 2005.8

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

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

目次

本書について

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

IBM(R) WebSphere(R) Business Integration Adapter ポートフォリオは、先進の 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・システムを統合的に接続する機能を提供します。本製品には、コンポーネントをカスタマイズ、作成、および管理するためのツールとテンプレートが含まれており、これにより、ビジネス・プロセスの統合を実現します。

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

対象読者

本書は、WebSphere Business Integration システムをお客様のサイトでサポートおよび管理する、コンサルタント、開発者、およびシステム管理者を対象としています。

本書の前提条件

本書の読者は、WebSphere Business Integration システム、ビジネス・オブジェクトとコラボレーションの開発、および JD Edwards OneWorld テクノロジーについて十分な知識と経験を持っている必要があります。

関連資料

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

資料は、次のサイトでダウンロード、インストール、および表示することができます。

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

一連の資料は PDF 形式のファイルおよび HTML 形式のファイルで構成されています。これらの資料を読むには、Netscape Navigator (バージョン 4.7 以降) や Internet Explorer (バージョン 5.5 以降) などの HTML ブラウザーと、Adobe Acrobat

Reader (バージョン 4.0.5 以降) が必要です。ご使用のプラットフォームに適した Adobe Acrobat Reader の最新バージョンについては、Adobe Web サイト (http://www.adobe.com) をご覧ください。

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

表記上の規則

本書は下記の規則に従って編集されています。

`Courier フォント`	コマンド名、ファイル名、入力情報、システムが画面に出力した情報など、記述されたとおりの値を示します。
太字	初出語を示します。
italic、*italic*	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、オプショナル・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意味します。
< >	命名規則では、不等号括弧は名前の個々の要素を囲み、各要素を区別します。(例: `<server_name><connector_name>tmp.log`)
/, ¥	本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。i5/OS では、ディレクトリー・パスにスラッシュ (/) を使用します。すべての製品のパス名は、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。
`%text%` および `$text`	% 記号で囲まれたテキストは、Windows の `text` システム変数またはユーザー変数の値を示します。
ProductDir	製品のインストール先ディレクトリーを表します。各プラットフォームのデフォルトは、以下のとおりです。 Windows: IBM¥WebSphereServer i5/OS: /QIBM/ProdData/WBIServer43/product i5/OS の場合、Capacity Pack Adapter は /QIBM/ProdData/WBIServer43/AdapterCapacityPack に存在します。

本リリースの新機能

リリース 2.0.x の新機能

- Adapter for JD Edwards OneWorld バージョン 2.0.x は、XML リスト API をサポートします。
- アダプターは Windows 2003 プラットフォーム (Standard Edition または Enterprise Edition) でサポートされます。
- アダプターは、WebSphere Business Integration Adapter Framework (管理ツールのみ) の場合は Windows XP Service Pack 1A でサポートされます。
- アダプターは OS/400 V5R2 および i5/OS V5R3 プラットフォームでサポートされます。

第 1 章 概要

本章では、WebSphere Business Integration Adapter for JD Edwards OneWorld のコネクター・コンポーネントの概要を説明します。以下のセクションに分かれています。

用語

このガイドでは、以下の用語が使用されています。

- **ASI (アプリケーション固有の情報)** 特定のアプリケーションまたはテクノロジーに合わせて作成されたメタデータ。ASI は、ビジネス・オブジェクトの属性レベル、動詞レベル、およびビジネス・オブジェクト・レベルに存在します。**動詞 ASI** も参照してください。
- **BF (ビジネス関数)** 特定のタスクの実行用に論理的にグループ化された、C 関数および関連データ構造の集合。レギュラー・ビジネス関数は、税額計算やアカウント番号検証などの単純なタスクを実行します。マスター・ビジネス関数は、さらに複雑なタスクを実行するもので、レギュラー・ビジネス関数をいくつか呼び出すことができます。
- **BO (ビジネス・オブジェクト)** ビジネス・エンティティー (Employee など) およびデータへのアクション (create または update 操作など) を表す属性のセット。WebSphere Business Integration システムのコンポーネントは、ビジネス・オブジェクトを使用して情報を交換し、アクションを起動します。
- **BO (ビジネス・オブジェクト) ハンドラー**・アプリケーションと対話し、要求ビジネス・オブジェクトをアプリケーションのオペレーションに変換するメソッドを格納するコネクター・コンポーネント。
- **コネクター・エージェント** 統合ブローカーからのサービス呼び出し要求および OneWorld からのイベント通知を処理するコネクターのコンポーネント。
- **接続オブジェクト** 接続とは、状態情報を格納することができるアプリケーションへの参照です。アダプター側の接続のすべてのインスタンスについて、JD Edwards OneWorld 側に対応するオブジェクトが存在します。ビジネス・オブジェクト・ハンドラーは必要に応じて、pool size プロパティーで指定された最大サイズまでの接続を作成します。新規接続はプールで維持され、複数のビジネス・オブジェクトの実行で再利用されます。
- **接続プール** 接続オブジェクトの保管と検索に使用されるリポジトリー。
- **GenJava** JD Edwards OneWorld が提供するユーティリティー。OneWorld サーバーの一部として実行されるビジネス関数のための Java ラッパーを生成します。GenJava は、インターフェース・クラスおよび関連データ構造の Java クラス・

ファイルを作成し、生成された Java ファイルをコンパイルし、Java 文書を生成し、それらを 2 つの JAR ファイルにパッケージします。1 つは Java クラス用、もう 1 つは Java 文書用です。

- **インターオペラビリティー・フレームワーク** さまざまに異なるソフトウェア・アプリケーション間で機能と情報のシームレスな共有を実現する。大小のビジネス関数にアクセスする単一のポイントを提供するビジネス関数ラッパーが含まれています。マスター・ビジネス関数ラッパーも含まれています。
- **Java オブジェクト** OneWorld ビジネス関数およびデータ構造を囲む、Java でインプリメントされたラッパー。Java オブジェクトは、OneWorld ビジネス関数と 1 対 1 対応しています。
- **ODA (Object Discovery Agent)** アプリケーション内部の指定されたエンティティーを調べ、ビジネス・オブジェクト属性に対応するこれらのエンティティーの要素を「検出」することによって、自動的にビジネス・オブジェクト定義を生成するツール。アダプターをインストールすると、ODA も自動的にインストールされます。
- **動詞 ASI (アプリケーション固有の情報)** 指定された動詞について、その動詞がアクティブであるときにコネクターがビジネス・オブジェクトを処理する方法を指定する。現在の要求ビジネス・オブジェクトを処理するために呼び出すメソッドの名前を格納することもできます。
- **XML リスト** OneWorld との検索インターフェースであり、これを使用して、テーブルまたは定義済みのテーブル変換プロセスからデータを取り出すことができます。

コネクターの概要

Connector for JD Edwards OneWorld は、WebSphere Business Integration Server Express Adapter for JD Edwards OneWorld のランタイム・コンポーネントです。コネクターは、OneWorld 8.0 (バージョン 7.3.3.4) および OneWorld 9.0 (PeopleSoft 8.9) をサポートします。

このアダプターは、同期式であり、完全に Java ベースです。JD Edwards Adapter は、OneWorld Java API を介して OneWorld と対話します。このアダプターには、OneWorld トリガーを介したイベント通知機構が組み込まれています。

JD Edwards OneWorld Adapter には、コネクター、メッセージ・ファイル、構成ツール、および Object Discovery Agent (ODA) が含まれています。統合ブローカーは、コネクターによって、ビジネス・オブジェクトと、OneWorld サーバーで実行されている対応する OneWorld オブジェクトの間で、データを交換することができます。

汎用 OneWorld アダプターの基本的な役割は、OneWorld サーバーと統合ブローカーの間の通信およびデータ交換を容易にするエージェントとして機能することです。アダプターは Java で開発されており、OneWorld が提供する GenJava インターフェース・ツールによって生成された OneWorld コンポーネント JAR ファイルを使用します。

OneWorld オブジェクトは、OneWorld サーバーの一部として実行されるビジネス関数です。WebSphere Business Integration Server Express Adapter for OneWorld は、OneWorld Java コネクターを使用してビジネス関数を起動します。

XML リスト API は、検索ビジネス関数と関連付けられていないビジネス・オブジェクトに対して検索機能を提供します。

コネクター・アーキテクチャー

コネクターは、2 つのコンポーネントで構成されています。コネクター・フレームワークおよびアプリケーション固有のコンポーネントです。コネクター・フレームワークは統合ブローカーとアプリケーション固有のコンポーネントの間の仲介役として機能し、そのコードはどのコネクターにも共通です。アプリケーション固有のコンポーネントには、特定のテクノロジー (この場合は JD Edwards OneWorld) またはアプリケーションに合わせて作成されたコードが含まれます。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントの間で、以下のサービスを提供します。

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

本書では、コネクター・フレームワークとアプリケーション固有のコンポーネントの両方について説明します。ここでは、これらの両方のコンポーネントを「コネクター」と呼んでいます。

Connector for JD Edwards OneWorld は、統合ブローカー InterChange Server Express とともに作動します。詳細については、「*システム・インプリメンテーション・ガイド*」を参照してください。

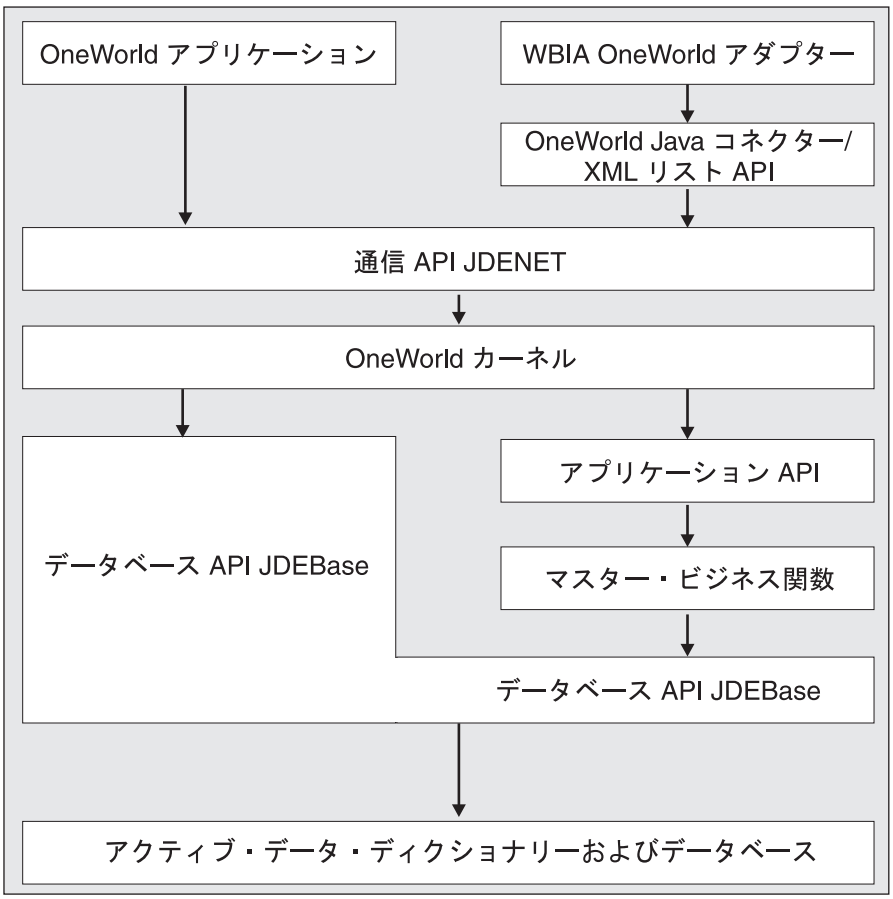

図 1. コネクター・アーキテクチャー

ビジネス関数

OneWorld ビジネス関数は、ジャーナル記入トランザクション、減価償却費の計算、および販売注文トランザクションなどの、特定のタスクを実行します。2 種類のビジネス関数があります。レギュラー・ビジネス関数は、税額計算やアカウント番号検証などの単純なタスクを実行します。マスター・ビジネス関数は、さらに複雑なタスクを実行するもので、レギュラー・ビジネス関数をいくつか呼び出して、それらのタスクを実行することができます。

インターオペラビリティー・フレームワークには、大小のビジネス関数にアクセスする単一のポイントを提供するビジネス関数ラッパーが含まれています。マスター・ビジネス関数ラッパーも含まれています。

特定のビジネス・オブジェクトの呼び出しに使用できるビジネス関数がない場合は、XML リストを使用できます。以下のダイアグラムは、XML 呼び出しのフローを示しています。

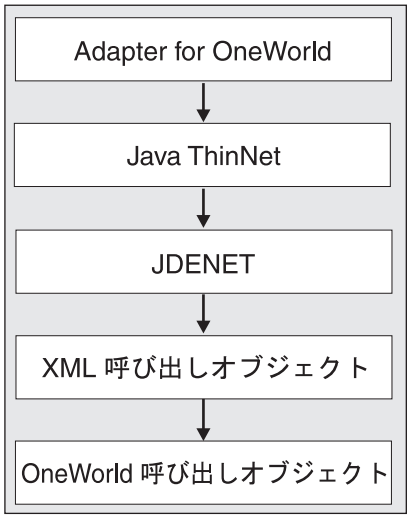

図 2. XML リスト呼び出しのフロー

I

XML 呼び出しが発生すると、以下のステップが発生します。

1. インターオペラビリティー・クライアントが、OneWorld に XML 文書を送信します。
2. クライアントが、C++ または Java ThinNet に定義されている API を使用して JDENET に XML 文書を送信します。
3. ThinNet が、マルチスレッド・アーキテクチャーを使用して、ロード・バランシングを行い、複数の XML 文書を同時に管理します。
4. インターオペラビリティー・クライアントが呼び出しオブジェクト (サービス呼び出しの処理などの同期要求でのみ使用される) を送信した場合、アダプターは XML トランザクション API を使用しません。Adapter for OneWorld から受信した XML 文書が、JDENET を使用して要求を処理します。以下のステップは、このプロセスを説明しています。
   a. XML 文書がソケット接続を作成する。
   b. JDENET メッセージを生成する。
   c. JDENET メッセージを送信する。
   d. JDENET メッセージを受信する。
   e. 応答データをアンパックする。
   f. ソケット接続をクローズする。
   g. XML 応答ファイルに応答データを渡す。
   h. 生成された応答ファイルを Adapter for OneWorld に戻す。
5. インターオペラビリティー・クライアントが呼び出しオブジェクトを送信しない場合、このクライアントは XML トランザクション API (通常は非同期要求に使用) を送信します。

以下のダイアグラムは、`XMLCallObject` で使用される ThinNet のプロセス・フローを示しています。

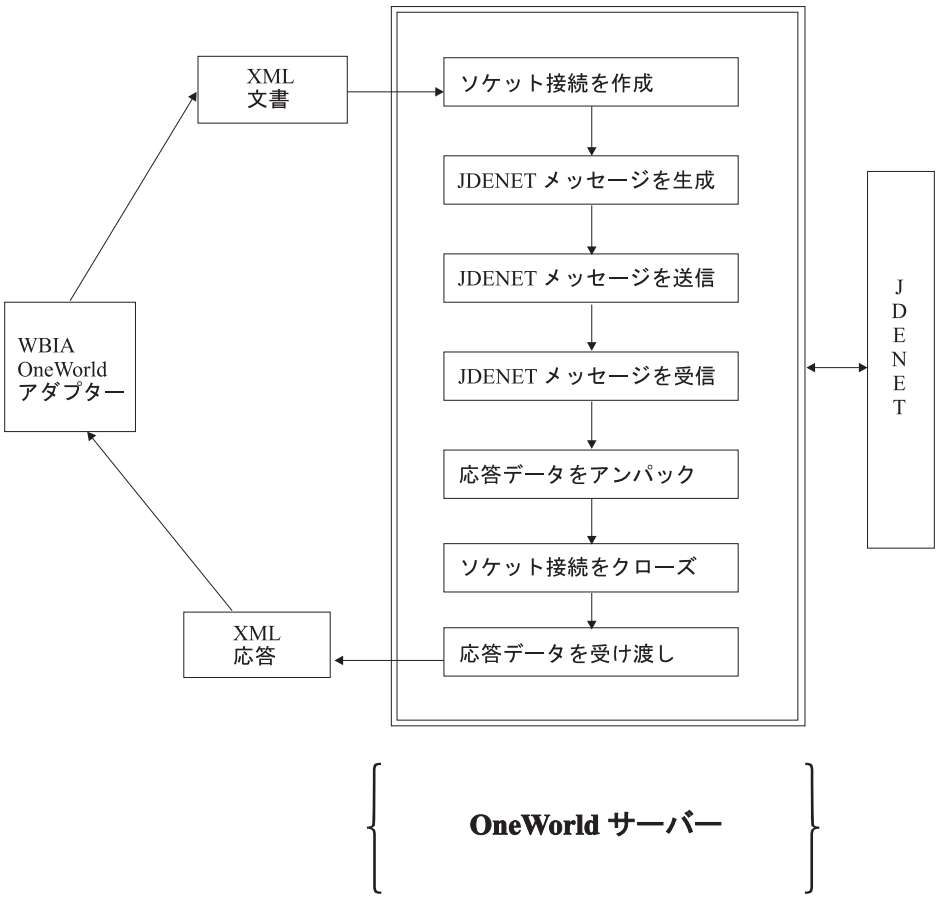

図 3. プロセス・フロー

以下の図は、アダプター・アーキテクチャーを示したものです。

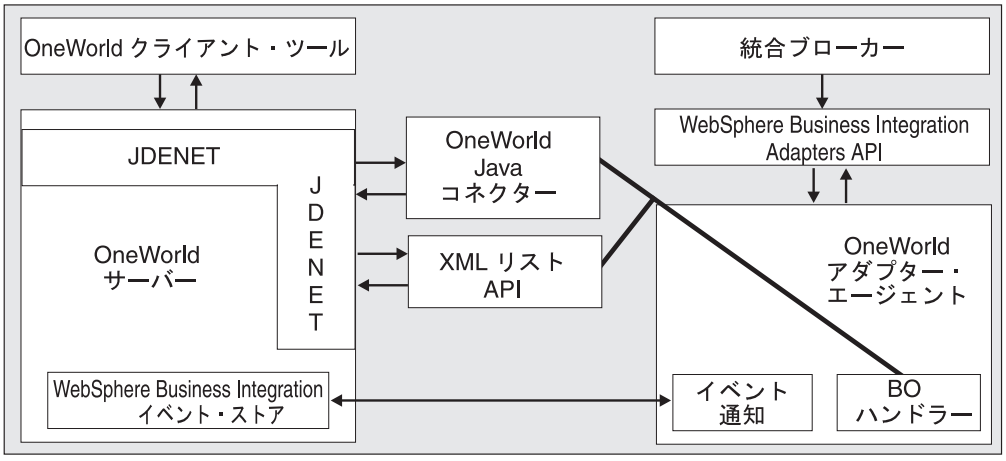

図 4. アーキテクチャーの図

OneWorld は、Java、COM、XML、および Table Conversion などのサード・パーティーのアプリケーションと通信するための API をいくつかサポートしています。

アダプターは Java API を使用して OneWorld 内のビジネス関数を起動します。ビジネス・オブジェクトは、ビジネス関数クラスまたはオブジェクトにマップされます。

実装の作業についてまとめると、以下のようになります。

1. GenJava プロセスの実行用に iJDEScript ファイルを作成します。
2. GenJava ユーティリティーを実行して、OneWorld オブジェクトの JAR ファイルを生成します。
3. ODA ツールを実行し、ビジネス関数および XML リスト用のビジネス・オブジェクトを生成します。 30 ページの『Business Object Designer Express の実行』を参照してください。
4. 2 つのビジネス・オブジェクト間でメタデータをマップする必要がある場合は、キー・フィールドおよび外部キー・フィールドを設定します。
5. アダプター構成ファイルにビジネス・オブジェクトを追加します。85 ページの『付録 B. Connector Configurator Express』を参照してください。
6. アダプターを始動します。 23 ページの『コネクターの始動』を参照してください。

要求処理

コネクター・フレームワークは、ブローカーから要求を受け取ると、要求ビジネス・オブジェクトのビジネス・オブジェクト定義と関連したビジネス・オブジェクト・ハンドラー・クラスの `doVerbFor()` メソッドを呼び出します。`doVerbFor()` メソッドの役割は、要求ビジネス・オブジェクトのアクティブな動詞に基づいて、実行する動詞の処理を決定することにあります。要求ビジネス・オブジェクトから情報を取得して、オペレーションの要求を作成し、アプリケーションへ送信します。

コネクター・フレームワークが要求ビジネス・オブジェクトを `doVerbFor()` に渡すと、このビジネス・オブジェクトがインターフェース・オブジェクトにマップされている場合、ビジネス・オブジェクト・ハンドラーが動詞 ASI を読み込み、それを一連の呼び出し可能な関数に変換します。これらの関数には、Business Object Designer Express で稼働している Object Discovery Agent (ODA) から、特定のセマンティックを与えることができます。ODA を使用してメソッド呼び出しシーケンスを動詞に割り当てる方法の詳細については、 29 ページの『第 4 章 ビジネス・オブジェクトの作成および変更』を参照してください。オブジェクトの処理を正常に行うためには、呼び出す順序が非常に重要になります。

動詞 ASI がブランクのインターフェース・ビジネス・オブジェクトの場合、ビジネス・オブジェクト・ハンドラーは、取り込んだパラメーターでビジネス関数属性を検索し、そのビジネス関数を呼び出します。1 つのメソッドのみにデータを取り込むことができます。そうではなく、複数のメソッドにデータが取り込まれているが、動詞 ASI はブランクであるという場合には、コネクターはエラーをログに記録して `FAIL` コードを戻します。エラー処理の詳細については、 57 ページの『エラー処理』を参照してください。

ビジネス・オブジェクトがビジネス関数オブジェクトにマップされている場合、ビジネス・オブジェクト・ハンドラーは、そのビジネス・オブジェクトで指定されたデータを持つ特定のビジネス関数を呼び出します。

特定のビジネス・オブジェクトに使用できる検索ビジネス関数がない場合は、検索機能のために XML リスト API を使用できます。

コネクターは、インターフェース・ビジネス・オブジェクトの特定の動詞はサポートしませんが、ビジネス・オブジェクトの動詞は、ODA を使用して構成できます。WebSphere Business Integration Server Express で使用される標準の動詞は、Create、Retrieve、Update、および Delete です。

ビジネス関数ビジネス・オブジェクトには、ODA がデフォルトの動詞 Execute を生成します。これらのビジネス・オブジェクトには、動詞 ASI は必要ありません。

ビジネス・オブジェクトの特殊なアクセス権をサポートするため、ACCESS_LEVEL という名前のメタ・ビジネス・オブジェクトが定義されています。ACCESS_LEVEL ビジネス・オブジェクトには、Username および Password という 2 つの属性があり、どちらもタイプはストリングです。特殊なアクセス規則を持ち、コネクター構成ファイルで指定された Username によってアクセスできない OneWorld ビジネス・オブジェクトは、このビジネス・オブジェクト (ACCESS_LEVEL) を単一カーディナリティーの子ビジネス・オブジェクトとします。この子ビジネス・オブジェクトは、doVerbFor 呼び出し内でトップレベル・ビジネス・オブジェクトのみに追加する必要があります。このトップレベル・ビジネス・オブジェクトのすべての子ビジネス・オブジェクトは、ACCESS_LEVEL 子ビジネス・オブジェクト内で指定された Username を通じてアクセス可能でなければなりません。

ビジネス・オブジェクト・ハンドラーは、トップレベル・ビジネス・オブジェクトにタイプ ACCESS_LEVEL の子ビジネス・オブジェクトがあるかどうかを検査します。そのような子ビジネス・オブジェクトがあり、そのビジネス・オブジェクト内の Username 属性の値がアダプターの使用している値と異なる場合は、新規の接続を開いて、子ビジネス・オブジェクトの属性 Username および Password の値を使用してビジネス・オブジェクトを処理します。ビジネス・オブジェクトの処理が完了してから、接続を閉じます。

トップレベル・ビジネス・オブジェクトがタイプ ACCESS_LEVEL の子オブジェクトを持たない場合、または UserName 属性がアダプター・プロパティーで指定されている UserName と同じ場合は、ビジネス・オブジェクト・ハンドラーはプールから接続オブジェクトを取り出します。

使用可能な接続オブジェクトが存在しない場合、プール・サイズが最大値に達していなければ、ビジネス・オブジェクト・ハンドラーは新規の接続オブジェクトをプール内に作成します。使用可能な接続オブジェクトが存在せず、プール・サイズが最大値に達している場合は、ハンドラーは使用可能になるまで待機します。

アダプターは、OneWorld インターフェース・クラスまたは XML リスト・ビジネス・オブジェクトにマップされるオブジェクトとして、トップレベル・ビジネス・オブジェクトをサポートします。アダプターは、トップレベル・ビジネス・オブジェクトとしてビジネス関数にマップされるビジネス・オブジェクトもサポートします。アダプターは、ビジネス・オブジェクトのタイプおよび構造に基づいて、ビジネス・オブジェクトを処理します。ODA により生成されるビジネス・オブジェクトに完全に依存する代わりに、論理表現を提供するためにビジネス関数にマップされるビジネス・オブジェクトを使用して、独自の階層を作成できます。例えば、Order の作成と Order Item の作成を行うビジネス関数を階層としてモデル化できます。

OrderItems を作成するビジネス関数にマップされるビジネス・オブジェクトは、Order ビジネス・オブジェクトを作成するためにマップされるビジネス・オブジェクトの子ビジネス・オブジェクトとなります。

アダプターは、ビジネス関数にマップされるトップレベル・ビジネス・オブジェクトを実行します。ビジネス・オブジェクトが子ビジネス・オブジェクトを持っていない場合、アダプターはビジネス・オブジェクトに対応するビジネス関数を実行します。このようなビジネス・オブジェクトの階層がビジネス・オブジェクト・ハンドラーへの入力である場合、アダプターは、1 つのトランザクションですべてのビジネス関数を実行します。この場合、動詞 ASI はブランクであり、ビジネス関数のフローはトップレベル・ビジネス・オブジェクトの属性の順序によって決定されます。例えば、トップレベル・ビジネス・オブジェクトが B110031 にマップされており、子 B110032 および B110033 を持っている場合、実行の順序は B110031、B110032、B110033 です。

ビジネス・オブジェクトの type 属性が XMLList の場合、ビジネス・オブジェクト・ハンドラーはビジネス・オブジェクトで定義された値とフォーマットを持つ XML 文書を作成します。アダプターは、XMLRequest オブジェクトを使用して、OneWorld に XML 文書を送信します。アダプターは応答を XML 応答文書として受信し、ハンドラーは XML 応答文書の応答データを使用してビジネス・オブジェクトにデータを取り込みます。

場合によっては、1 つの呼び出しシーケンスで、単一の単純属性またはオブジェクトを複数回使用しなければならないことがあります。属性 ASI を使用すると、2 つの属性をリンクすることができます。属性が外部キーとしてマークされている場合、この属性は 1 つの属性 ASI である use_attribute_value= タグを持っている必要があり、この値は BusinessObject.AttributeName に対応している必要があります。このリンクは、ソース・ビジネス・オブジェクトが単一カーディナリティーである場合にのみ使用してください。複数カーディナリティーを持つソース・ビジネス・オブジェクトを使用して構成されている場合、アダプターはリストの最初のビジネス・オブジェクトを選択し、そのビジネス・オブジェクトの値をマップします。

アプリケーション・イベントの処理

イベント通知には、アダプターとともに出荷されるイベント・パッケージ BIA_EVENT のインストールと、JDE データベースへのイベント表およびアーカイブ表の作成が必要です。

注: iSeries OS/400 V5R2 または i5/OS V5R3 プラットフォームで JD Edwards アプリケーション・サーバーを実行している場合は、必要なイベント・パッケージの使用可能性について IBM ソフトウェア・サポートに問い合わせてください。

イベント・パッケージ BIA_EVENT のインストールおよび構成の方法の情報については、25 ページの『IBM イベント・ストアのインストールおよび構成』を参照してください。

JD Edwards OneWorld アプリケーションにおけるレコードの作成、更新、または削除の操作は、イベントとして処理できます。イベント表の取り込みには、OneWorld

でサポートされるテーブル・トリガーを使用できます。イベント表にイベントを生成するには、JD Edwards が推奨するその他のメソッドを使用することもできます。これらのレコードは、pollForEvents の呼び出し時に取得され、処理されます。イベントに関連するイベント・テーブル・ストアの情報は、 49 ページの表 5 で説明しています。

注: Event ID は、イベント表内で一意である必要があります。

注: イベントがサブスクライブされている間に、コネクターは、表 5 の情報を使用して対応するビジネス・オブジェクトをビルドし、後続処理のためにそれらのオブジェクトをコネクター・フレームワークに送信します。

イベント処理用ビジネス・オブジェクトの検索

イベント処理のためのオブジェクトの検索は、キー属性と非キー属性の両方に基づいて行います。ビジネス・オブジェクトが JD Edwards のビジネス関数を表す場合には動詞 Execute が、ビジネス・オブジェクトがインターフェースを表す場合には動詞 Retrieve がサポートされることが必須です。

イベント管理

コネクターは IBM イベント表 (F5501005) を一定間隔でポーリングしてイベントを検索し、最初は優先順位順に、次に順次にイベントを処理します。コネクターがイベントを処理すると、イベントの状況がそれに応じて更新されます。

ArchiveProcessed プロパティーを設定すると、イベントの状況を更新後、コネクターが IBM アーカイブ表 (F5501006) にイベントをアーカイブするかどうかが決定します。ArchiveProcessed プロパティーの詳細については、 19 ページの『コネクター固有のプロパティー』を参照してください。 20 ページの表 2 は、
ArchiveProcessed プロパティーの設定に基づいたアーカイブの振る舞いを示しています。

コネクターの処理

このセクションでは、コネクターがクライアントとして実行される場合に、コネクターのさまざまな部分がビジネス・オブジェクトを処理する方法を説明します。

1. コネクターの始動時、コネクターのエージェント・クラスは以下の初期化 (Init) 処理を実行します。
   - 構成プロパティーを検索します。
   - コネクター構成ファイルから Username、Password、および Environment を取り出します。
   - OneWorld コネクター・オブジェクトを作成します。
   - Login メソッドと、取り出した Username および Password を用いたパラメーターを使用して、OneWorld サーバーにログインします。このメソッドは、SessionID を戻します。
   - OneWorld インターフェース・オブジェクトのインスタンスを作成します。
   - コネクター、OneWorldInterface、および SessionID を接続プールに追加します。

2. OneWorld ビジネス・オブジェクト・ハンドラーは、動詞 ASI を読み取り、一連の呼び出し可能な関数または子オブジェクトに変換します。
   - ビジネス・オブジェクトにタイプ ACCESS_LEVEL の子ビジネス・オブジェクトがあり、この子ビジネス・オブジェクト内の Username 属性にデータが取り込まれていて、アダプターが使用する値と異なる値になっている場合、ビジネス・オブジェクト・ハンドラーは、ACCESS_LEVEL ビジネス・オブジェクトに指定された Username 属性および Password 属性の値を使用して新規の接続を開きます。すべてのこのようなビジネス・オブジェクトで、Username 属性および Password 属性の両方にデータが取り込まれている必要があります。
   - アプリケーションがダウンしているために接続の作成が失敗した場合、ビジネス・オブジェクト・ハンドラーは APPRESPONSETIMEOUT を戻します。
   - Username/Password が不正なために接続の作成が失敗した場合、ビジネス・オブジェクト・ハンドラーはエラーをログに記録し、FAIL の状況を戻します。
   - ビジネス・オブジェクトがタイプ ACCESS_LEVEL の子ビジネス・オブジェクトを持たないか、またはこのビジネス・オブジェクト内の Username 属性が null であるかアダプターの Username に指定された値と同じである場合には、使用可能な接続プールから接続を取り出します。ビジネス・オブジェクト・ハンドラーが使用可能な接続を要求したときに、接続プールで行われる処理を以下に示します。
     a. ビジネス・オブジェクト・ハンドラーは、プール内に使用可能な接続があるかどうかを検査します。
     b. 使用可能な接続がある場合には、接続の有効性をチェックします。接続が無効な場合、ハンドラーは接続の再作成を試行します。
     c. 接続の作成が失敗した場合は、APPRESPONSETIMEOUT 状況を戻します。
     d. ビジネス・オブジェクト・ハンドラーは、その接続を使用可能リストから除去し、使用中リストに追加します。
     e. 接続が使用不可であり、接続の最大数がプール・サイズよりも小さい場合には、新規の接続を開き、接続プールの使用中リストに追加します。新規接続のオープンが失敗した場合、アダプターは APPRESPONSETIMEOUT を戻します。
     f. 使用可能な接続が存在せず、プール・サイズの最大限度に達している場合には、doVerbFor スレッドは接続が使用可能になるまで待機します。
   - ビジネス・オブジェクトのタイプが BFN の場合、アダプターは以下のアクションを実行します。
     a. アダプターは、OneWorld クラス OneWorldInterface の BeginTransaction メソッドを使用してトランザクションを開始します。
     b. ビジネス・オブジェクトがインターフェース・クラスにマップされ、動詞 ASI がブランクである場合、アダプターは、ビジネス・オブジェクト内に取り込まれている最初のメソッド属性または最初の子オブジェクトを検出し、実行します。
     c. 動詞 ASI にデータが取り込まれている場合、アダプターは InvokeMethods を呼び出します。これにより、動詞 ASI で指定されたすべてのメソッドをループ処理します。
     d. ビジネス・オブジェクトがビジネス関数にマップされる場合、呼び出し側はビジネス・オブジェクトにマップされるビジネス関数を実行します。タ

イプが ACCESS_LEVEL ではない子オブジェクトが存在する場合、ビジネス・オブジェクト・ハンドラーはそれらをループ処理し、トップレベル・ビジネス・オブジェクト内で定義された順序で、それらに対応するビジネス関数を実行します。

e. 呼び出し側は、ビジネス・オブジェクト内で定義された属性に基づいて引数を構成してから、リフレクション API を使用して OneWorld Java オブジェクト上でメソッドを起動します。

f. ビジネス・オブジェクト全体の実行が正常に終了すると、ビジネス・オブジェクト・ハンドラーはオブジェクト OneWOrldInterface 上で Commit メソッドを使用してトランザクションをコミットし、VALCHANGED 状況を戻します。

- ビジネス・オブジェクトのタイプが XMLList の場合、アダプターは以下のアクションを実行します。
  a. アダプターは、ビジネス・オブジェクトに指定されている値およびフォーマットを持つ XML 文書を作成します。
  b. アダプターは、XML リスト API を使用して、この文書を OneWorld に送信します。
  c. 障害が発生した場合、アダプターは応答文書にエラー・コードおよび理由を記録します。応答文書に問題があった場合、アダプターはログ・ファイルにも戻り状況 FAIL でエラーを記録します。
  d. アダプターが OneWorld に XML 文書を正常に送信できた場合、応答文書の値がビジネス・オブジェクトで更新されます。
  e. ビジネス・オブジェクトがタイプ ACCESS_LEVEL でない子ビジネス・オブジェクトを持っている場合、ハンドラーはそれぞれの子ビジネス・オブジェクトごとに前述のステップを繰り返します。
  f. アダプターがビジネス・オブジェクト全体を正常に処理すると、状況が VALCHANGED に設定されます。
- 接続を接続プールに解放します。
- ビジネス関数が正常に実行されたときは VALCHANGED を戻します。
- ビジネス・オブジェクトのタイプが BFN で、Interface クラスにマップされているとき、動詞 ASI がブランクになっており、データを取り込んだ属性が無い場合は、FAIL を戻します。
- 処理が失敗した場合は FAIL を戻します。

3. ConnectionEventStore クラスは、サブスクリプション・デリバリーで以下の処理を実行します。
   - コネクターがイベントを受信すると、ConnectionEventStore クラスは以下の処理を行います。
     – イベントが指定するタイプのビジネス・オブジェクトを作成します。
     – イベント表が指定するオブジェクト・キーを使用して、そのビジネス・オブジェクトのキーの値とキー以外の値を設定します。
     – ビジネス・オブジェクトのタイプがビジネス関数の場合、動詞に Execute を設定します。
     – ビジネス・オブジェクトのタイプがインターフェースの場合、動詞に Retrieve を設定します。

- ビジネス・オブジェクトの取得後、コネクターは、イベントが指定する動詞を設定してビジネス・オブジェクトを統合ブローカーに送信します。

4. 接続プールから取得したすべての接続をクローズして終了します (Terminate)。

第 2 章 アダプターのインストール

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 OneWorld バージョン 2.0.0 は、以下のアダプター・フレームワークと統合ブローカーでサポートされます。

- **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、2.4 および 2.6。
- **統合ブローカー**:
  - InterChange Server Express

前提事項とサード・パーティー依存

JD Edwards OneWorld 対応のコネクターをインストールする前に、以下の前提事項およびソフトウェア要件を検討してください。

プラットフォーム要件

コネクターは、以下のプラットフォーム上で実行されます。

- Windows 2003 (Standard Edition または Enterprise Edition)
- WebSphere Business Integration Adapter Framework (管理ツールのみ) の場合は Windows XP Service Pack 1A
- OS/400 V5R2 および i5/OS V5R3。明示的に述べられていない限り、i5/OS は OS/400 および i5/OS を参照します。

Adapter for JD Edwards OneWorld のアダプターと関連ファイルのインストール

WebSphere Business Integration Server Express アダプター製品のインストールについては、次のサイトで WebSphere Business Integration Server Express Adapters インフォメーション・センターにある「*Installation Guide for Windows and OS/400*」を参照してください。

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

WebSphere Business Integration Adapter のディレクトリーおよびファイル

インストールが完了すると、ファイル・システムおよびその内容を表示できます。作成されるフォルダーやファイルは、インストール時の選択およびオペレーティング・システムによって異なります。

インストーラーは、コネクターに関連付けられた標準ファイルをご使用のシステムにコピーします。Windows の場合、コネクター・エージェントを *ProductDir*¥connectors¥OneWorld ディレクトリーにインストールして、コネクター・エージェントへのショートカットを「スタート」メニューに追加します。i5/OS の場合は、ショートカットが WebSphere Business Integration Console に自動的に追加されます。

ProductDir は、アダプター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。環境変数には、*ProductDir* ディレクトリーへのパス (デフォルトでは *IBMWebSphereAdapters*) が含まれています。

コネクターのファイル構造

インストーラーは、コネクターに関連付けられた標準ファイルをご使用のシステムにコピーします。

ユーティリティーによって、コネクターが *ProductDir*¥connectors¥OneWorld ディレクトリーにインストールされ、コネクターのショートカットが「スタート」メニューに追加されます。*ProductDir* は、製品がインストールされているディレクトリーを表していることに注意してください。

表 1 には、コネクターが使用するファイル構造が記載されており、インストーラーを介したコネクターのインストールを選択した際に自動的にインストールされるファイルを示します。

表 1. コネクターのファイル構造

***ProductDir* のサブディレクトリー**	**説明**
¥connectors¥OneWorld¥BIA_OneWorld.jar	OneWorld コネクターのみに使用されるクラスを含む。
¥connectors¥OneWorld¥start_OneWorld.bat	汎用コネクター (Windows) の始動スクリプト。
/connectors/OneWorld/start_OneWorld.sh	汎用コネクター (i5/OS) の始動スクリプト。
¥connectors¥OneWorld¥dependencies¥BIA_IBMEvents.cmd	IBM イベント・ストア・ビジネス関数のスクリプト・ファイルが格納されます。このファイルは、GenJava プロセスを実行してイベント・ストア・ビジネス関数の Java ラッパーを作成する際に使用できます。
¥connectors¥OneWorld¥dependencies¥BIA_EVENT.exe	eventstore パッケージをインストールする実行可能モジュール (OneWorld 8.0 をサポート)。
¥connectors¥messages¥OneWorldAdapter.txt	コネクターのメッセージ・ファイル。
¥ODA¥OneWorld¥BIA_OneWorldODA.jar	OneWorld ODA。
¥ODA¥OneWorld¥start_OneWorldODA.bat	ODA 始動ファイル (Windows)
/ODA/OneWorld/BIA_OneWorldODA.sh	ODA 始動ファイル (i5/OS)
¥ODA¥messages¥BIA_OneWorldODAAgent_de_DE.txt	ODA 用のメッセージ・ファイル (ドイツ語のテキスト・ストリング)。
¥ODA¥messages¥BIA_OneWorldODAAgent_en_US.txt	ODA 用のメッセージ・ファイル (米国英語のテキスト・ストリング)。
¥ODA¥messages¥BIA_OneWorldODAAgent_es_ES.txt	ODA 用のメッセージ・ファイル (スペイン語のテキスト・ストリング)。
¥ODA¥messages¥BIA_OneWorldODAAgent_fr_FR.txt	ODA 用のメッセージ・ファイル (フランス語のテキスト・ストリング)。
¥ODA¥messages¥BIA_OneWorldODAAgent_it_IT.txt	ODA 用のメッセージ・ファイル (イタリア語のテキスト・ストリング)。
¥ODA¥messages¥BIA_OneWorldODAAgent_ja_JP.txt	ODA 用のメッセージ・ファイル (日本語のテキスト・ストリング)。
¥ODA¥messages¥BIA_OneWorldODAAgent_ko_KR.txt	ODA 用のメッセージ・ファイル (韓国語のテキスト・ストリング)。
¥ODA¥messages¥BIA_OneWorldODAAgent_pt_BR.txt	ODA 用のメッセージ・ファイル (ポルトガル語 (ブラジル) のテキスト・ストリング)。

表 1. コネクターのファイル構造 (続き)

ProductDir のサブディレクトリー	説明
¥ODA¥messages¥BIA_OneWorldODAAgent_zh_CN.txt	ODA 用のメッセージ・ファイル (中国語 (簡体字) のテキスト・ストリング)。
¥ODA¥messages¥BIA_OneWorldODAAgent_zh_TW.txt	ODA 用のメッセージ・ファイル (中国語 (繁体字) のテキスト・ストリング)。
repository¥OneWorld¥BIA_CN_OneWorld.txt	コネクターのリポジトリー定義。デフォルト名は BIA_OneWorld.txt です。
¥connectors¥OneWorld¥start_OneWorld_service.bat	コネクター・サービスの始動スクリプト。
connector\OneWorld\dependencies\BIA_EVT9.exe	eventstore パッケージをインストールする実行可能モジュール (OneWorld9.0 PeopleSoft 8.9.2 をサポート)。
/dependencies/jdeinterop.ini	操作不能にするためのサーバー設定ファイル。

注: すべての製品のパス名は、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。

インストール後の作業

Adapter for JD Edwards OneWorld を正常にインストールしたら、以下のインストール後の処理を実行してください。

- 『アダプターの構成』
- 『ファイルのコピー』
- 18 ページの『ODBC 接続の作成』

アダプターの構成

アダプターのインストール後にアダプターを初めて始動する前に、アダプターを構成する必要があります。詳細については、 19 ページの『第 3 章 コネクターの構成』を参照してください。

ファイルのコピー

GenJava のインストールの間に、アダプターで使用されるビジネス関数クラスが含まれる .jar ファイルが作成されます。これらの .jar ファイルを ProductDir¥connectors¥OneWorld¥repository フォルダーにコピーしてください。アダプターおよび ODA が、これらのファイルを動的にアップロードします。

イベント通知コンポーネントによって使用されるビジネス関数の Java ラッパーを作成するには、アダプターと共に提供される BIA_IBMEvents.cmd ファイルを使用して GenJava を実行します。生成された .jar ファイルを ProductDir¥connectors¥OneWorld¥dependencies フォルダーにコピーしてください。

OneWorld 8.0

B7334¥System¥classes フォルダーにある以下のファイルを ProductDir¥connectors¥OneWorld¥dependencies フォルダーにコピーしてください。

- Kernal.jar
- Connector.jar

i5/OS の場合は、ファイルを /QIBM/UserData/WBIServer43/instance name/connectors/OneWorld/dependencies にコピーします。ここで instance_name は InterChange Server Express インスタンスの名前です。

jdeinterop.ini ファイルを B7334¥Interoperability¥Examples¥Java フォルダーから ProductDir¥connectors¥OneWorld¥dependencies にコピーしてください。jdeinterop.ini ファイルの編集方法については、「*Interoperability Guide*」を参照してください。

OneWorld 9.0

B9¥System¥classes フォルダーにある以下のファイルを ProductDir¥connectors¥OneWorld¥dependencies フォルダーにコピーしてください。

- Kernal.jar
- Connector.jar
- Logic.jar
- Jdeutil.jar
- Database.jar

サンプルの jdelog.properties ファイルと jdeinterop.ini.templ ファイルを jdeinterop.ini ファイルと同様に B9¥System¥classes¥Samples フォルダーから ProductDir¥connectors¥OneWorld¥dependencies フォルダーにコピーしてください。

jdelog.properties ファイルと jdeinterop.ini ファイルを編集します。フィルターのセクションの編集方法については、「*OneWorld Interoperability Guide*」を参照してください。

ODBC 接続の作成

コネクターを始動およびプルするために、OneWorld アダプターに各 DB2 UDB データベースの ODBC データ・ソースが必要です。ODBC 接続の作成方法の情報については、JD Edwards の「*Installation Guide*」を参照してください。

第 3 章 コネクターの構成

コンポーネントをインストール後、開始する前に、このセクションで説明するようにコンポーネントを構成する必要があります。

- 23 ページの『コネクターの始動』
- 25 ページの『コネクターの停止』
- 26 ページの『ログ・ファイルとトレース・ファイルの使用』

コネクターの構成プロパティーには、標準とアダプター固有という 2 つのタイプがあります。アダプターを実行する前に、Connector Configurator Express を使用してこれらのプロパティーの値を設定する必要があります。詳細については、85 ページの『付録 B. Connector Configurator Express』を参照してください。

コネクターは、始動時に構成値を取得します。ランタイム・セッション中に、1 つ以上のコネクター・プロパティーの値の変更が必要になることがあります。一部のコネクター構成プロパティーへの変更は、即時に有効になります。その他のコネクター・プロパティーへの変更を有効にするには、変更後にコネクター・コンポーネントまたはシステムを再始動する必要があります。プロパティーが動的 (即時に有効になる) であるか静的 (コネクター・コンポーネントまたはシステムの再始動が必要) であるかを判別するには、System Manager の 「コネクター・プロパティー」ウィンドウ内の「更新メソッド」列を参照してください。

i5/OS の場合、JDEInterop.ini ファイルはコネクターが動作するために、i5/OS 固有の値で編集する必要があります。

標準コネクター・プロパティー

標準コネクター構成プロパティーにより、すべてのアダプターによって使用される情報が提供されます。標準構成プロパティーの資料については、61 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、これらの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ビルドを行わなくても、コネクター内の静的な情報またはロジックを変更できます。

コネクター固有のプロパティーを構成するには、Connector Configurator Express を使用します。「**アプリケーション構成プロパティー**」タブをクリックして、構成プロパティーを追加または変更します。詳しくは、85 ページの『付録 B. Connector Configurator Express』を参照してください。

20 ページの表 2 に、コネクターに対するコネクター固有の構成プロパティーを示し、その説明と指定可能な値も示します。プロパティーの詳細については、以下の各セクションを参照してください。

表 2. コネクター固有のプロパティー

名前	指定可能な値	デフォルト値
Username	JDE	なし
Password	JDE	なし
PoolSize	5	5
Environment	DV7334	なし
ServerName	JDEDEV1	なし
PortNo	6010	なし
PollQuantity	任意の数値。	1
EventStoreFactory	com.ibm.adapters.oneworld.OneWorldEventStoreFactoryInstance	com.ibm.adapters.oneworld.OneWorldEventStoreFactoryInstance
InDoubtEvents	Reprocess、Fail on startup、Log error、Ignore	Ignore
ArchiveProcessed	True、False	True
UseDefaults	True	True

Username

OneWorld アプリケーションに接続するためのユーザー名。これは必須プロパティーです。

Password

OneWorld アプリケーションに接続するためのパスワード。これは必須プロパティーです。

PoolSize

接続プール内の接続の最大数。この数は、JDEInterop.ini ファイルで指定されている接続の最大許可数を超えないようにする必要があります。これは必須プロパティーです。i5/OS の場合、JDEInterop.ini ファイルは i5/OS 固有の値で編集する必要があります。

Environment

OneWorld 内の、接続を実行すべき環境の名前。これは必須プロパティーです。

ServerName

OneWorld サーバーを実行しているマシンの名前。これは、XML リスト・ビジネス・オブジェクトを使用する場合は必須プロパティーです。

PortNo

OneWorld サーバーが listen しているポート番号。これは、XML リスト・ビジネス・オブジェクトを使用する場合は必須プロパティーです。

PollQuantity

それぞれのポーリング・サイクルごとにアプリケーションから取り出されるイベントの数。

EventStoreFactory

このプロパティーは、イベント・ストア・ファクトリー・インスタンス・クラスの名前です。値は `com.ibm.adapters.omeworld.OneWorldEventStoreFactoryInstance` です。

InDoubtEvents

不完全なイベントを処理するかどうかを決定します。デフォルト値は `Ignore` です。

ArchiveProcessed

デフォルト値が設定されているかどうかをチェックします。`ArchiveProcessed` が `true` に設定されている場合、アーカイブ用のビジネス関数がイベントをアーカイブ表に保存します。このプロパティーが `false` に設定されているか、または設定されていない場合は、アーカイブ用ビジネス関数は呼び出されません。デフォルト値は `true` です。

UseDefaults

デフォルト値が設定されているかどうかを検査します。デフォルト値は `true` です。これは必須プロパティーです。

複数コネクター・インスタンスの作成

コネクターの複数インスタンスの作成は、多くの点でカスタム・コネクターの作成と似ています。以下に示すステップを実行することによって、コネクターの複数のインスタンスを作成して実行するように、ご使用のシステムを設定することができます。それには、以下の作業を行う必要があります。

- コネクター・インスタンスの新規ディレクトリーを作成する
- 必要なビジネス・オブジェクト定義が存在することを確認する
- 新規コネクター定義ファイルを作成する
- 新規始動スクリプトを作成する

新規ディレクトリーの作成

- **Windows プラットフォームの場合**:

  ProductDir¥connectors¥connectorInstance

  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は、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ここに格納します。

  ProductDir¥repository¥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。

InterChange Server Express サーバー名を startup.bat のパラメーターとして指定できます。
例えば、start_XML.bat connName serverName です。

- **i5/OS プラットフォームの場合:**

  /QIBM/UserData/WBIServer44/WebShereICSName/connectors
  /connectorInstance

  ここで、connectorInstance はコネクター・インスタンスを固有に識別し、WebSphereICSName はコネクターが実行する Interchange Server Express インスタンスの名前です。

  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は、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ここに格納します。

  /QIBM/UserData/WBIServer44/WebSphereICSName/repository
  /connectorInstance。
  ここで WebSphereICSName はコネクターの実行に使用する Interchange Server Express インスタンスの名前です。

ビジネス・オブジェクト定義の作成

プロジェクト内にコネクター・インスタンスごとのビジネス・オブジェクト定義が存在しない場合は、ビジネス・オブジェクト定義を作成する必要があります。

1. 初期コネクターに関連付けられているビジネス・オブジェクト定義を変更する必要がある場合は、適切なファイルをコピーし、Business Object Designer Express を使用してそれらのファイルをインポートします。初期コネクターには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` の該当する `connectorInstance` サブディレクトリーに存在します。

コネクター定義の作成

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

1. 初期コネクターの構成ファイル (コネクター定義) をコピーし、ファイルを名前変更します。
2. 各コネクター・インスタンスがサポートしているビジネス・オブジェクト (および関連メタオブジェクト) が正しくリストされていることを確認します。
3. 適宜コネクター・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 22 ページの『ビジネス・オブジェクト定義の作成』で作成したコネクター・ディレクトリーに置きます。
3. (Windows のみ) 始動スクリプトのショートカットを作成します。
4. (Windows のみ) 初期コネクターのショートカット・テキストをコピーして、新規コネクター・インスタンス名に一致するように初期コネクターの名前を (コマンド行で) 変更します。
5. (i5/OS のみ) 次の情報を使用して、コネクターのジョブ記述を作成します。CRTDUPOBJ(QWBIEMAILC) FROMLIB(QWBISVR44)OBJTYPWE(*JOBD) TOLIB (QWBISVR44) NEWOBJ(newconnname) ここで、newconnname は新規のコネクターのジョブ記述に使用する 10 文字の名前です。
6. (i5/OS のみ) WebSphere Business Integration Server Express Console に新規のコネクターを追加します。WebSphere Business Integration Server Express コンソールの詳細については、コンソールに付属するオンライン・ヘルプ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。例えば、Windows では ProductDir¥connectors¥connName を使用します。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オペレーティング・システムにより異なります。Windows でコネクターを開始するには、`start_OneWorld.bat` スクリプトを実行します。i5/OS の場合は、`start_OneWorld.sh` スクリプトを実行します。

Windows での始動スクリプトの起動

Windows プラットフォームでは、以下の方法でコネクターの始動スクリプトを起動できます。

- System Monitor から、このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。
- 「スタート」メニューから
  - **「プログラム」>「IBM WebSphere Business Integration Express」>「アダプター」>「コネクター」>** *「ご使用のコネクター名 (your_connector_name)」*を選択します。
    デフォルトでは、プログラム名は「IBM WebSphere Business Integration Server Express」となっています。ただし、これはカスタマイズすることができます。あるいは、ご使用のコネクターへのデスクトップ・ショートカットを作成することもできます。

– コネクターは、Windows システムの Windows サービスとして始動するように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。

- コマンド行から `start_connName connName WebSphereICSName [-cconfigFile ]` と入力します。
  ここで、*connName* はコネクターの名前で、*WebSphereICSName* は InterChange Server Express インスタンスの名前です。デフォルトでは、InterChange Server Express インスタンスの名前は WebSphereICS です。

i5/OS での始動スクリプトの起動

i5/OS プラットフォームでは、以下の方法でコネクターの始動スクリプトを起動できます。

- System Monitor から、このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。
- WebSphere Business Integration Console がインストールされている Windows から、「**プログラム**」>「**IBM WebSphere Business Integration Console**」>「**コンソール**」を選択します。次に、i5/OS システム名または IP アドレスと、`*JOBCTL` 特殊権限を持つユーザー・プロファイルおよびパスワードを指定します。アダプターのリストから connName アダプターを選択し、「**アダプターを始動**」ボタンを選択します。
- i5/OS コマンド行から
  – **バッチ・モード:** CL コマンド QSH を実行し、QSHELL 環境から `/QIBM/ProdData/WBIServer43/bin/submit_adapter.sh connName WebSphereICSName pathToConnNameStartScript jobDescriptionName` を実行します。ここで、*connName* はコネクター名、*WebSphereICSName* は InterChange Server Express サーバー名 (デフォルトは QWBIDFT)、*pathToConnNameStartScript* はコネクター始動スクリプトへの絶対パス、*jobDescriptionName* は QWBISVR43 ライブラリーで使用するジョブ記述の名前です (デフォルトのジョブ記述は QWBIJDEC)。
  – **対話モード:** CL コマンド QSH を実行し、QSHELL 環境から `/QIBM/UserData/WBIServer43/WebSphereICSName/connectors /connName/start_connName.sh connName WebSphereICSName [-cConfigFile]` を実行します。ここで、*connName* はコネクターの名前で、*WebSphereICSName* は Interchange Server Express インスタンスの名前です。

注: TCP/IP サーバーで始動するには、次のコマンドを使用します。
`/QIBM/ProdData/WBIServer43/bin/add_autostart_adapter.sh connName WebSphereICSName pathToConnNameStartScript jobDescriptionName`。ここで、*connName* はコネクター名、*WebSphereICSName* は InterChange Server Express サーバー名 (デフォルトは QWBIDFT)、*pathToConnNameStartScript* はコネクター始動スクリプトの絶対パス、*jobDescriptionName* は QWBISVR43 ライブラリーで使用するジョブ記述の名前です (デフォルトのジョブ記述は QWBIJDEC)。

コネクターの停止

コネクターを停止する方法は、コネクターが始動された方法によって異なります。

Windows からのコネクターの停止

Windows プラットフォームでは、以下の方法でコネクターを停止できます。

- System Monitor から、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。
- 「コネクター」ウィンドウをアクティブにして、**q** と入力して **Enter** を押します。
- コネクターが Windows のサービスとして始動された場合は、コントロール・パネルを使用してコネクターを停止できます。

i5/OS からのコネクターの停止

i5/OS プラットフォームでは、以下の方法でコネクターを停止できます。

- System Monitor から、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。
- WebSphere Business Integration Console またはコマンド行から、コンソールを使用して、または i5/OS コマンド入力から QSHELL で *submit_adapter.sh* スクリプトを使用してコネクターを始動した場合は、CL コマンド WRKACTJOB SBS(QWBISVR43) を使用して Server Express 製品に対するジョブを表示します。リストをスクロールして、コネクターのジョブ記述と一致するジョブ名のジョブを見つけます (JD Edwards の場合、ジョブ名のデフォルトは QWBIJDEC です)。このジョブに対してオプション 4 を選択し、**F4** を押して ENDJOB コマンドのプロンプトを取得します。次に、オプション・パラメーターとして *IMMED を指定し、**Enter** を押します。
- QSHELL から *start_connName.sh* スクリプトを使用してアダプターを始動した場合は、**F3** を押してコネクターを終了します。

IBM イベント・ストアのインストールおよび構成

アダプター・パッケージに、実行可能ファイル BIA_EVENT.exe ファイルが含まれています。この実行可能ファイルは、IBM イベント・ストア・コンポーネントをインストールします。JD Edwards はこれをソフトウェア更新として参照します。ソフトウェア更新の適用については、JD Edwards の指示に従ってください。イベント・ストア (ソフトウェア更新) コンポーネントは、イベント表とアーカイブ表、およびイベント表とアーカイブ表のイベントを取得、削除、更新、およびアーカイブするのに使用するビジネス関数で構成されています。

イベント処理のビジネス関数、テーブル定義ファイル、およびデータ項目は、イベント・パッケージ BIA_EVENT.exe の一部です。配置サーバーまたは配置ワークステーションを準備してからソフトウェア更新を実行し、その後、以下の更新手順に従ってください。JD Edwards の 「*Software Update Installation Guide*」で、準備および更新の手順を詳細に説明しています。

BIA_EVENT.exe ファイルは、必要なビジネス関数とテーブル定義スクリプトを作成しますが、JD Edwards クライアントを使用して、イベント表とアーカイブ表を作成

する必要があります。配置サーバーまたは配置ワークステーションへのソフトウェア更新を正常にインストールしたら、アダプターがイベント・ストアを検出できるようにエンタープライズ・サーバーにコンポーネントを配置する必要があります。

BIA_EVENTパッケージの内容は以下のとおりです。

- B551005 -- Retrieve_WBIAEvents
- B551006 -- Update_WBIAEvent
- B551007 -- Archive_WBIAEvent
- B551008 -- Delete_WBIAEvent
- B551009 -- Recover_WBIAEvent

以下がテーブル定義ファイルです。

- F5501005.h -- イベント表構造
- F5501006.h -- アーカイブ表構造

以下のデータ項目が含まれています。

- EVENT_ID
- EVT_DESC
- EVT_PRTY
- EVT_STATUS
- EVT_TIME
- ADAPTER_ID
- ARCHIVE_T
- OBJ_KEY
- OBJ_NAME
- OBJ_VERB

注: パッケージにはその他のデータ項目も含まれますが、テーブルでは使用されないため、次のリリースではパッケージから除去される予定です。

ログ・ファイルとトレース・ファイルの使用

アダプター・コンポーネントは、いくつかのレベルのメッセージ・ロギングおよびトレースを提供します。コネクターは、アダプター・フレームワークを使用してエラー・メッセージ、通知メッセージ、およびトレース・メッセージをログに記録します。エラー・メッセージおよび通知メッセージは、ログ・ファイルに記録され、トレース・メッセージおよび対応するトレース・レベル (0 から 5) は、トレース・ファイルに記録されます。ロギングおよびトレース・レベルの詳細については、57 ページの『第 6 章 エラー処理』を参照してください。

Connector Configurator Express 内で、ログ・ファイル名およびトレース・ファイル名のほか、トレース・レベルも構成します。このツールの詳細については、85 ページの『付録 B. Connector Configurator Express』を参照してください。

ODA からのエラー・メッセージは、ODA 用に指定されたトレース・ファイル名のついたファイルに記録されます。不正なトレース・ファイルが指定されていると、

メッセージは ODA が実行されている画面プロンプトに送信されます。トレース・ファイルおよびトレース・レベルは Business Object Designer Express で構成します。このプロセスについては、 31 ページの『エージェントの構成』で説明します。ODA トレース・レベルは、 58 ページの『トレース』で定義されているコネクター・トレース・レベルと同じです。

第 4 章 ビジネス・オブジェクトの作成および変更

本章では、Object Discovery Agent (ODA) for JD Edwards OneWorld について説明し、それを使用して Adapter for JD Edwards OneWorld 対応のビジネス・オブジェクト定義を生成する方法を説明します。

本章の内容は、次のとおりです。

ODA for OneWorld の概要

ODA (Object Discovery Agent) を用いて、ビジネス・オブジェクト定義を生成することができます。ビジネス・オブジェクト定義は、ビジネス・オブジェクトのテンプレートです。ODA は、指定されたアプリケーション・オブジェクトを検査し、ビジネス・オブジェクト属性に対応するオブジェクトの要素を「検出」し、また、情報を表すためのビジネス・オブジェクト定義を生成します。Business Object Designer Express には、Object Discovery Agent にアクセスしたり、そのエージェントと対話式に作業を行ったりするためのグラフィカル・インターフェースが用意されています。

Object Discovery Agent (ODA) for JD Edwards OneWorld は、GenJava に生成される .jar ファイルからビジネス・オブジェクト定義を生成します。ビジネス・オブジェクト・ウィザードは、これらの定義を作成するプロセスを自動化します。ODA を使用してビジネス・オブジェクトを作成し、Connector Configurator Express を使用して、それらのオブジェクトをサポートするコネクターを構成します。Connector Configurator Express については、85 ページの『付録 B. Connector Configurator Express』を参照してください。

ビジネス・オブジェクト定義の生成

このセクションでは、Business Object Designer Express 内で JD Edwards OneWorld ODA を使用してビジネス・オブジェクト定義を生成する方法を説明します。Business Object Designer Express の起動および使用については、「*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

Windows での ODA の始動

ODA は、Windows の `start_OneWorld0DA.bat` で、メタデータ・リポジトリー (すなわち、IDL ファイル) が存在するファイル・システムをマウント可能な任意のマシンから実行できます。この始動ファイルには、必要な OneWorld .jar ファイルおよび Connector .jar ファイルへのパスなどの始動パラメーターが記述されています。

OneWorld 対応の ODA のデフォルト名は、OneWorldODA です。この名前は、始動スクリプト (start_OneWorldODA.bat) 内の AGENTNAME を変更することによって変更できます。

ODA を始動するには、次のコマンドを実行します。

```
start_OneWorldODA
```

i5/OS での ODA の始動

ODA を始動するには、
/QIBM/UserData/WBIServer43/<instance_name>/ODA/OneWorld/
start_OneWorldODA.sh の ODA 始動スクリプトを変更する必要があります。

JAR_DIR ディレクトリーを設定する行は、Kernel.jar と Connector.jar がコピーされた場所に変更する必要があります。

JD Edwards Oneworld からコピーされた JAR ファイルのパスを入力します。

i5/OS で ODA を始動するには、次のいずれかの方法を使用します。

- WebSphere Business Integration Console がインストールされている Windows システムから、「**プログラム**」>「**IBM WebSphere Business Integration Console**」>「**コンソール**」を選択します。次に、i5/OS システム名または IP アドレスと、*JOBCTL 特殊権限を持つユーザー・プロファイルおよびパスワードを指定します。ODA のリストから ODA を選択して、「**ODA を始動**」ボタンを選択します。
- i5/OS コマンド入力から QSH CL コマンドを実行し、QSHELL 環境から次のパラメーターの順で /QIBM/ProdData/WBIServer43/bin/submit_oda.sh スクリプトを実行します。
  pathToODAStartScript jobDescriptionName
  ここで、*pathToODAStartScript* は ODA 始動スクリプトの絶対パス、*jobDescriptionName* は QWBISVR43 ライブラリーで使用するジョブ記述の名前です。
- i5/OS コマンド入力から QSH CL コマンドを実行し、QSHELL コマンド入力から ODA 始動スクリプトを直接実行します。
  start_ODAName.sh

Business Object Designer Express の実行

Business Object Designer Express では、ODA を使用してビジネス・オブジェクト定義を生成するステップをユーザーに示すウィザードを提供しています。以下に示すステップがあります。

- 『エージェントの選択』
- 31 ページの『エージェントの構成』
- 33 ページの『ビジネス・オブジェクトの選択』
- 35 ページの『オブジェクト選択の確認』

エージェントの選択

1. Business Object Designer Express 始動します。

2. 「**ファイル**」>「**ODA を使用して新規作成**」 をクリックします。「*ビジネス・オブジェクト・ウィザード - ステップ 1/6 - エージェントの選択*」画面が表示されます。
3. 「**検索されたエージェント**」リストで ODA/AGENTNAME (start_OneWorldODA スクリプトから) を選択し、「**次へ**」をクリックします。(必要なエージェントがリストされていない場合は「**エージェントの検索**」 をクリックする必要があります。)

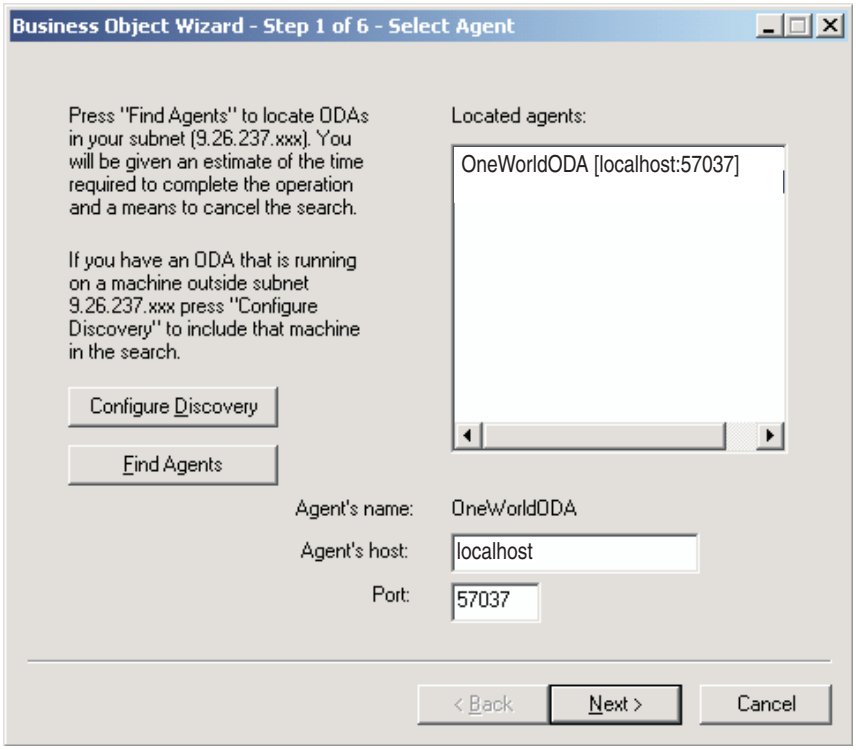

図 5. 「エージェントの選択」画面

エージェントの構成

「**次へ**」をクリックすると、「*ビジネス・オブジェクト・ウィザード - ステップ 2/6 - エージェントの構成*」画面が表示されます。 32 ページの図 6 では、この画面をサンプル値とともに示しています。

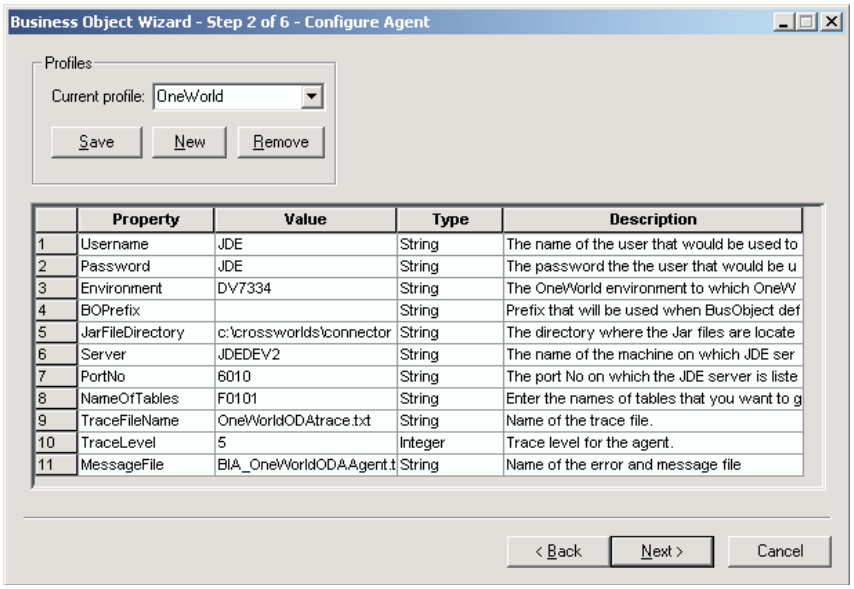

図 6. 「エージェントの構成」画面

この画面で設定するプロパティーを、表 3 で説明します。この画面で入力したすべての値をプロファイルに保管することができます。次回 ODA を実行するときに、プロパティー・データを再入力しなくても、ドロップダウン・メニューからプロファイルを選択するだけで、保管された値を再利用できます。それぞれ指定された値の組み合わせが異なる複数のプロファイルを保管することも可能です。

表 3. 「エージェントの構成」プロパティー

プロパティー名	デフォルト値	タイプ	説明
BOPrefix	なし	String	ODA が、生成するビジネス・オブジェクトの名前の前に付加するプレフィックス。
Environment	なし	String	OneWorld への接続に使用する環境。
JarFileDirectory	なし	String	(必須) .jar ファイルが配置されているディレクトリー。アダプターを使用して起動する必要のあるビジネス関数を含む .jar ファイルはすべて、このディレクトリーに配置しなければなりません。
NameOfTables	なし	String	ビジネス・オブジェクトを生成する対象となる、; で区切られた OneWorld のテーブル名 (例: F4211;f4210)。
Password	なし	String	OneWorld への接続に使用するパスワード。
PortNo	なし	String	JDE サーバーが実行しているポート番号。
Server	なし	String	JDE が実行しているマシン名。
TraceFileName	なし	String	トレース・メッセージ・ファイルの名前。例えば OneWorldODAtrace.txt など。

表 3. 「エージェントの構成」プロパティー (続き)

プロパティー名	デフォルト値	タイプ	説明
TraceLevel	5	Integer	(必須) エージェントのトレース・レベル (0 から 5)。トレース・レベルの詳細については、 58 ページの『トレース』を参照してください。
MessageFile	なし	String	(必須) ODA が表示するすべてのメッセージを記述したメッセージ・ファイルの名前。OneWorld の場合、このファイルの名前は BIA_OneWorldODAAgent.txt となります。メッセージ・ファイルの名前を正しく指定しなかった場合、ODA はエラーを生成します。
UserName	なし	String	OneWorld への接続に使用するユーザー名。

1. ODA で新規プロファイルを作成したい場合は、任意の時点で、「プロファイル」グループ・ボックス内の「**新規**」ボタンおよび「**保管**」ボタンを使用します。ODA を再び使用するときには、既存のプロファイルを選択することができます。
2. 32 ページの表 3 に定義されているように、各プロパティーの値を入力します。

   注: プロファイルを使用する場合、プロパティー値はプロファイルから取り込まれますが、必要に応じて値を変更することができます。新規の値を保管することも可能です。

ビジネス・オブジェクトの選択

34 ページの図 7 に示すように、「*ビジネス・オブジェクト・ウィザード - ステップ 3/6 - ソースの選択*」画面が表示されます。

生成するオブジェクトの選択に関連する規則を以下にリストします。

- 親オブジェクトを選択すると、生成する子オブジェクトも自動的に選択されます。子オブジェクトとともに親オブジェクトを選択すると、選択した子オブジェクトのみが生成されます。
- 親を選択せずに子オブジェクトを選択すると、子オブジェクトは生成されますが、親オブジェクトは生成されません。
- すべての子ビジネス・オブジェクトは、単一カーディナリティーの包含関係で生成されます。
- ビジネス・オブジェクトが複数カーディナリティーとして振る舞う必要がある場合は、Business Object Designer Express を使用してカーディナリティーを手動で変更する必要があります。
- ODA はビジネス・オブジェクトのキー・フィールドにマークを付けないため、保管する前にビジネス・オブジェクト内のキー・フィールドに手動でマークを付ける必要があります。
- 生成する .jar ファイルを選択できます。これにより、.jar ファイルに含まれるすべてのインターフェース・ビジネス・オブジェクトおよびビジネス関数ビジネス・オブジェクトの定義が生成されます。

この画面にリストされる OneWorld オブジェクトの中で、上位オブジェクトの子オブジェクトであるものを判別するには、オリジナルの GenJava ファイルを参照してください。また、この画面にリストされるすべての OneWorld オブジェクトを選択して、それぞれに対応するオブジェクトを生成することもできます。生成されたビジネス・オブジェクトは、親子関係を反映します。

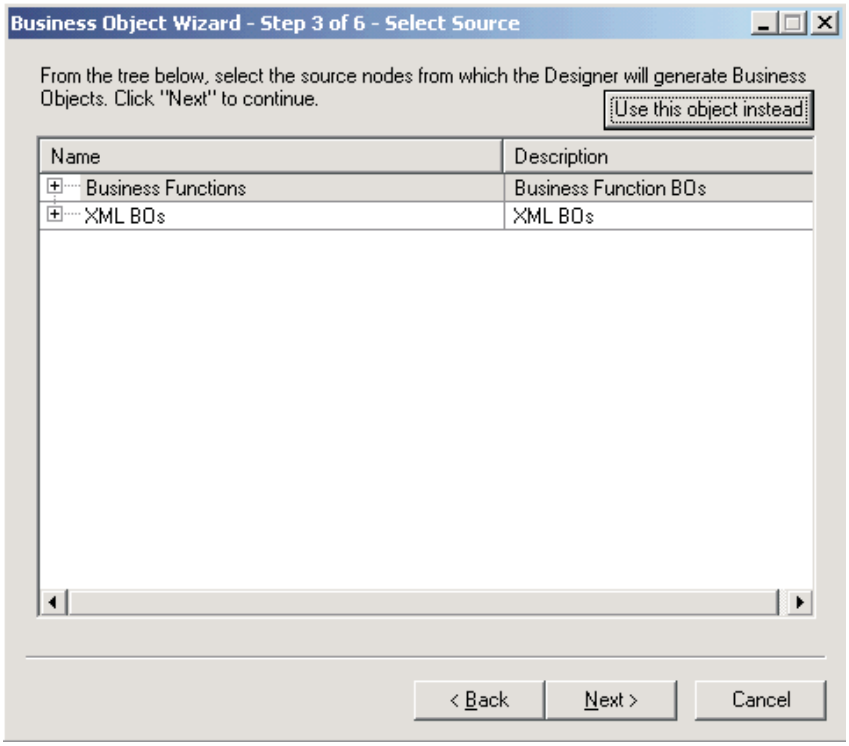

図 7. 「ソースの選択」画面

ODA により、2 つのオプション 「ビジネス関数 (Business Functions) 」および「XML BO (XML BOs)」がツリー・ノードとして表示されます。「ビジネス関数 (Business Functions)」をクリックすると、.jar ファイル形式の OneWorld オブジェクトを含むツリーが JarFileDirectory 構成プロパティーで指定されたディレクトリーの下に表示されます。「ビジネス関数 (Business Functions)」に固有の情報については、36 ページの『ソース・ノードの選択』を参照してください。

「XML BO (XML BOs)」をクリックすると、ツリーの子ノードが表示されます。これらのツリーの子ノードは、ODA のプロパティー・ウィンドウで入力したテーブルの名前に対応します。リストされたテーブルから、生成するテーブルを選択できます。「XML BO (XML BOs)」に固有の情報については、39 ページの『XML ビジネス・オブジェクト』を参照してください。

ビジネス・オブジェクトを生成するために、複数の .jar ファイルからオブジェクトを選択できます。オブジェクトを選択するには、「代わりにこのオブジェクトを使用」ボタンを使用します。標準のフィルター機能を使用すると、ツリーの子ノードの一部を選択できます。

1. 必要であれば、OneWorld モジュールを展開して、サブオブジェクトのリストを表示します。

2. 使用する OneWorld オブジェクトを選択します。
3. 「**次へ**」をクリックします。

オブジェクト選択の確認

「*ビジネス・オブジェクト・ウィザード - ステップ 4/6 - ビジネス・オブジェクト定義のソース・ノードの確認*」画面が表示されます。この画面には、選択したオブジェクトが表示されます。

ODA を用いて生成したビジネス・オブジェクトについては、ビジネス・オブジェクトを保管する前に、Business Object Designer Express 内でキー・フィールドに手動でマークを付ける必要があります。ODA は、属性をキー・フィールドとしてマーク付けすることはありません。あるビジネス・オブジェクトから別のビジネス・オブジェクトに値をマップすることを計画している場合は、外部キーにマークを付ける必要があります。ビジネス・オブジェクトの属性が、すでに処理済みのほかのビジネス・オブジェクトの値を必要とする場合は、ASI タグ `use_attribute_value` を属性 ASI に手動で追加する必要があります。

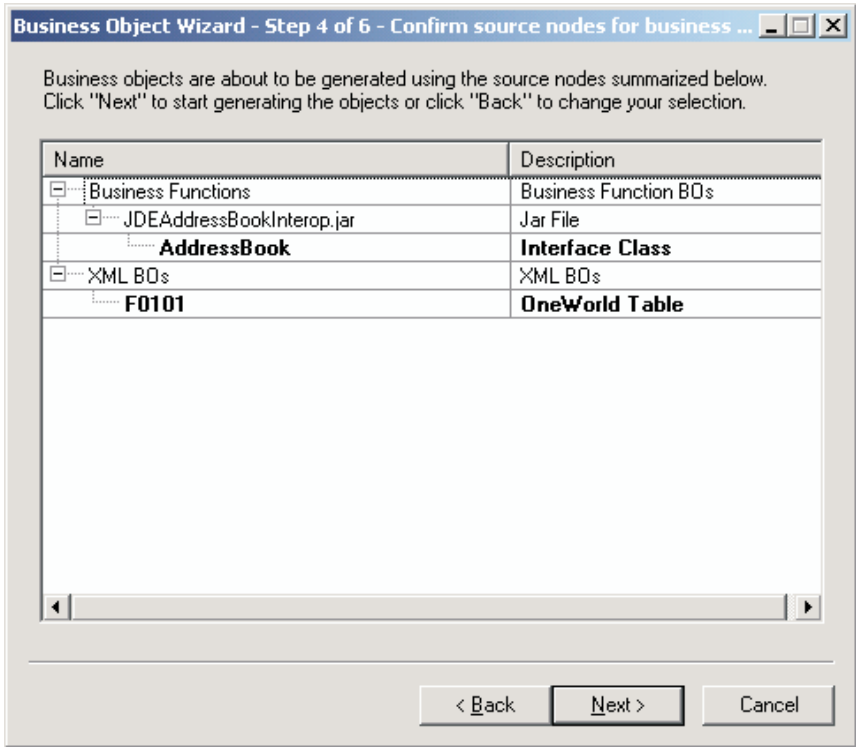

図 8. 「ソース・ノードの確認」画面

変更するには「**戻る**」をクリックし、リストが正しいことを確認するには「**次へ**」をクリックします。

「*ビジネス・オブジェクト・ウィザード - ステップ 5/6 - ビジネス・オブジェクトの生成中...*」画面が表示され、そこに、ウィザードがビジネス・オブジェクトを生成していることを示すメッセージが表示されます。

ソース・ノードの選択

ビジネス関数ビジネス・オブジェクトと XML リスト・ビジネス・オブジェクトの両方を、ODA の 1 回の実行で生成できます。両方の種類のビジネス・オブジェクトの並列生成がサポートされます。

ビジネス関数: ビジネス関数ビジネス・オブジェクトを生成する場合、ODA では、選択されたすべてのクラスをロードしてイントロスペクトすることにより、OneWorld オブジェクト用のビジネス・オブジェクトを生成します。ODA は「BO プロパティー」ウィンドウを使用して、各 OneWorld オブジェクト用のユーザーの構成情報を取得します。

インターフェース・クラスにマップされるビジネス・オブジェクトのためのビジネス・オブジェクト・プロパティーのウィンドウは 2 種類あります。最初の「取り込み中 (Capturing)」ウィンドウでは、選択されたすべての OneWorld オブジェクトに対してサポートされる動詞を取り込みます。2 番目の「動詞 ASI のメソッド・シーケンスを取り込み中 (Capturing Method sequence for Verb ASI)」ウィンドウでは、各ビジネス・オブジェクトのそれぞれの動詞ごとにメソッド・シーケンスを取り込みます。データ構造ビジネス・オブジェクトの場合は、1 つのデフォルトの動詞 Execute が作成されますが、この動詞には ASI はありません。

OneWorld オブジェクトは、属性およびメソッドを持つことができます。ビジネス・オブジェクト定義の名前は、OneWorld オブジェクト名から派生し、これにビジネス・オブジェクト・プレフィックスが付加されます。同じデータ構造クラスを使用するビジネス関数が 2 つある場合、ODA では、ビジネス・オブジェクト ASI 内のビジネス関数名を除いては同じ構造を持つ複数の定義を生成します。ビジネス・オブジェクトの名前は、_1、_2... というように増分します。例えば、2 つの関数が D0100033 という名前を使用している場合、生成されるビジネス・オブジェクトは D0100033、D0100033_1 です。

get/set メソッド

OneWorld データ構造オブジェクトの場合、get/set メソッドには対応するビジネス・オブジェクト属性があります。属性のタイプは type=<type> として ASI に保管され、属性の実際の名前は name=<name> として ASI に保管されます。1 つの get/set の組み合わせに対して、1 つの属性のみが生成されます。例えば、属性名が ID の場合、メソッドは getID() および setID() です。この場合、ビジネス・オブジェクトは、ID という名前で ASI が getter=getID();setter=setID() (type=int、name=ID) の 1 つの属性を持ちます。

ビジネス関数の属性

OneWorld のインターフェース Java クラスに定義されているそれぞれのビジネス関数ごとに、1 つの属性が作成されます。ビジネス関数呼び出しを適切に表現するため、この属性のタイプは、ビジネス関数呼び出しの入力パラメーターを表現する属性を含む子ビジネス・オブジェクトになります。属性の名前はビジネス・オブジェクトの名前になり、タイプはデータ構造ビジネス・オブジェクトの名前になります。属性は、タグ bfn_name= を使用してビジネス関数の名前を保持する ASI を持っています。

ビジネス関数のパラメーター

ビジネス関数の入力パラメーターは、属性として表現されます。あるパラメーターのデータ型が、WebSphere Business Integration フォーマットでサポートされないデータ型である場合、その型は「String」として表現されます。基本的に、パラメーターはデータ構造クラスに定義されている変数です。パラメーターの ASI には、元のデータ型の値と名前が保管されます。サポートされるパラメーターのリストについては、48 ページの『ビジネス・オブジェクト属性のタイプ』を参照してください。

OneWorld のオブジェクト・タイプ

GenJava を使用して生成される OneWorld の jar には、次の 2 種類の Java クラスがあります。

- インターフェース・クラス・ファイル

  インターフェース・クラス・ファイルはライブラリーにマップされ、インターフェースの組み合わせは `iJDEScript` ファイルに定義されています。このファイルには、すべてのビジネス関数および指定されたインターフェースにインポートされたビジネス関数の `create<businessfunction>ParameterSet` メソッドのシグニチャーが格納されています。

  ファイルの内容が以下のとおりであるとします。

  ```
  login
  library JDEAddressBook
     interface AddressBook
        import B0100031
        import B0100019
        import B0100032
        import B0100002
        import B0100033
  build
  logout
  ```

  この場合、`AddressBook.class` のクラス・ファイルは、B0100031 のメソッドおよび Create<B0100031>ParameterSet を持つことになります。ここで、<B0100031> はビジネス関数の英語のテキスト名です。

- データ構造クラス・ファイル

  データ構造クラス・ファイルは、特定のビジネス関数の入力として必要なすべてのパラメーターの get メソッドと set メソッドを保持します。

インターフェース・クラスにマップされるビジネス・オブジェクト: インターフェース・クラスにマップされるビジネス・オブジェクトのビジネス・オブジェクト情報を指定できます。ビジネス・オブジェクトの作成後に、そのオブジェクトで有効な動詞、オブジェクトでの指定された動詞のメソッド・シーケンス、ビジネス・オブジェクト・レベル ASI、および属性レベル ASI を指定することができます。このセクションでは、ODA を Business Object Designer Express とともに使用して、この情報を指定する方法を説明します。これらの情報のカテゴリーと、JD Edwards OneWorld コネクター内のビジネス・オブジェクト構造での役割の詳細については、45 ページの『第 5 章 ビジネス・オブジェクトの理解』を参照してください。

動詞の選択

選択されたビジネス・オブジェクトが OneWorld のインターフェース・オブジェクトにマップされている場合、Business Object Designer Express で、ビジネス・オブ

ジェクトの作成完了後に別個のウィンドウでそのビジネス・オブジェクトを開いたときに最初に表示される画面は、「*BO プロパティー - コンポーネントの動詞を選択してください*」画面です。図 9 は、AddressBook ビジネス・オブジェクトに対応するこの画面を示します。XML リストのビジネス関数にマップされるビジネス・オブジェクトの場合、単一の動詞 execute が作成されます。

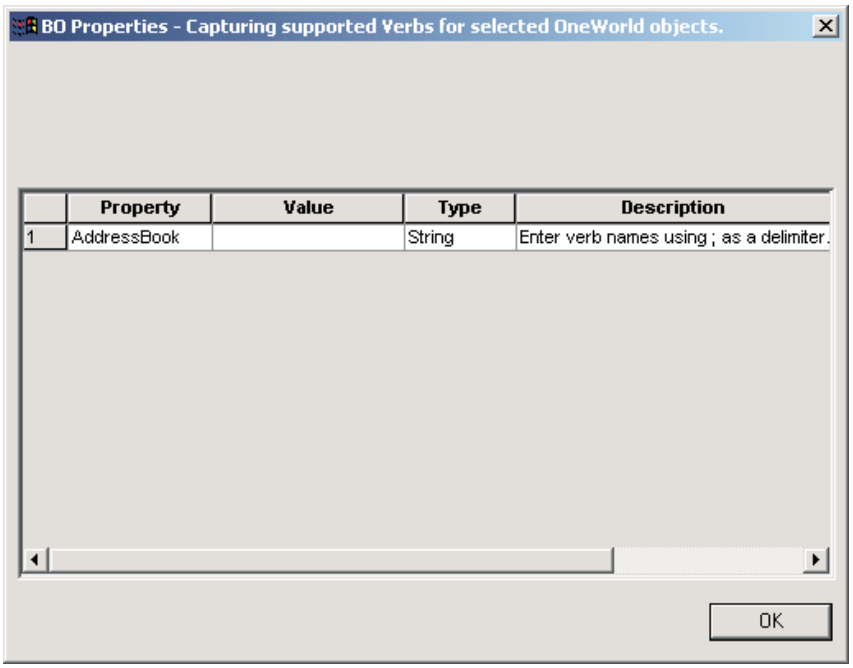

図 9. 「コンポーネントの動詞を選択してください」画面

この画面では、ビジネス・オブジェクトがサポートする動詞を指定できます。動詞の名前を入力してそれらを「;」で区切ると、特定のビジネス・オブジェクトに必要な動詞を指定できます。動詞の名前は「*ビジネス・オブジェクト開発ガイド*」で指定されている命名規則に合っている必要があります。

WebSphere Business Integration で使用される標準の動詞は、Create、Retrieve、Delete、および Update です。OneWorld コネクター用のビジネス・オブジェクト動詞 ASI の詳細については、50 ページの『動詞 ASI』を参照してください。

動詞 ASI の指定

選択した動詞ごとに、ビジネス関数が動詞を実行する指定の順序で、個別のウィンドウが表示されます。

39 ページの図 10 は、 34 ページの図 7 および 35 ページの図 8 で作成した、「ビジネス関数 (Business Functions)」の下の AddressBook ビジネス・オブジェクトの Retrieve 動詞の画面を示しています。

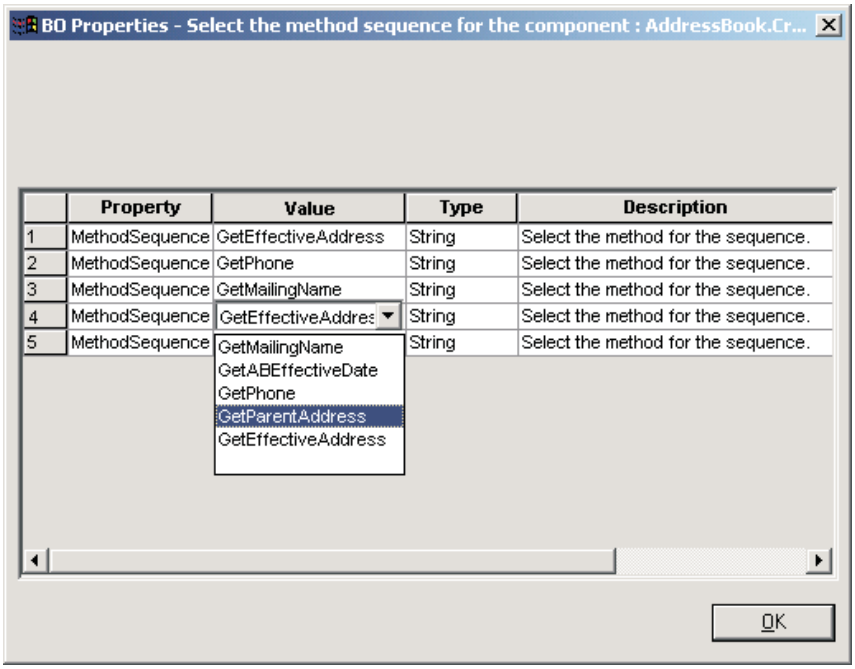

図 10. 動詞メソッド・シーケンスの設定

MethodSequence プロパティーの「**値**」リストから、その動詞に関してビジネス・オブジェクトに最初に実行させたいメソッドを選択できます。図 10 では、メソッド・シーケンスは以下のようになっています。

- Retrieve 動詞のメソッド・シーケンスで実行される最初のメソッドは GetEffectiveAddress です。
- シーケンスの 2 番目のメソッドは GetPhone です。
- シーケンスの 3 番目のメソッドは GetMailingName です。

動詞のビジネス関数シーケンスを指定すると、その動詞に関連付けられた動詞 ASI が作成されます。必要な場合、この動詞 ASI は後から変更することができます。

XML ビジネス・オブジェクト: 35 ページの図 8 に示したソース・ノードの確認の画面で選択したそれぞれのテーブルごとに、追加情報を取り込むための後続のプロパティー・ウィンドウが表示されます。

XML ビジネス・オブジェクトの場合は、次の 3 種類のビジネス・オブジェクト・プロパティーのウィンドウが表示されます。

- テーブル・タイプ: 選択されたすべての OneWorld テーブルについてテーブル・タイプを取り込みます。
- データの選択: それぞれのテーブルごとに、where 文節のプロパティーを選択できます。
- データの順序付け: 照会のデータの順序付けを選択できます。

テーブル・タイプ

ビジネス・オブジェクト・プロパティーのウィンドウには、プロパティー名とプロパティー値が表示されます。それぞれのテーブルごとに、テーブル・タイプを選択できます。ドロップダウン・メニューを使用して、単一カーディナリティーを選択できます。

次のプロパティー値から選択できます。

- OWTABLE
- OWVIEW
- FOREIGN_TABLE
- TABLE_CONVERSION

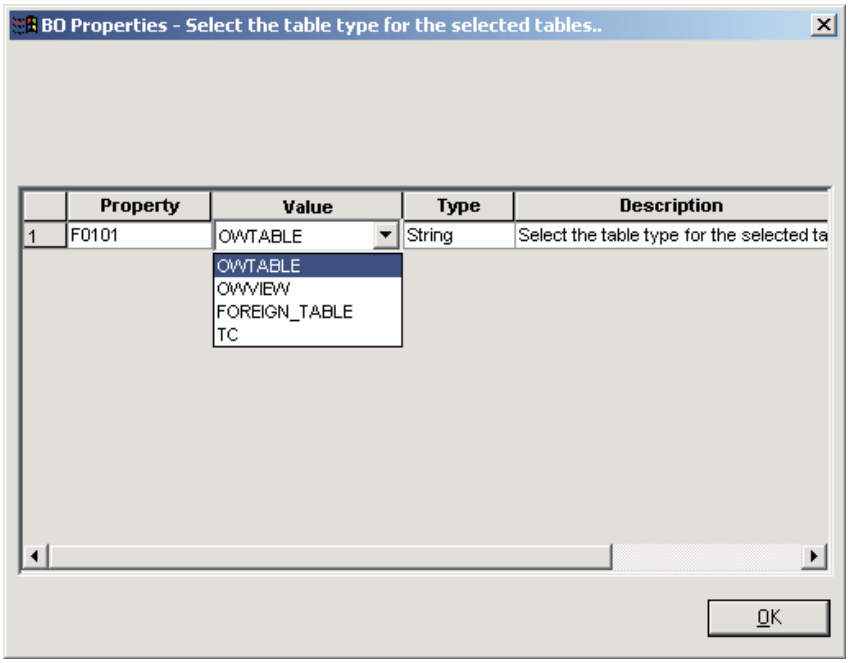

図 11. 「選択されたテーブルのテーブル・タイプを選択してください (Select the table type for selected tables)」画面

テーブル・タイプを選択すると、ODA により、それぞれのテーブルごとに GetTemplate API が呼び出されます。ODA は、GetTemplate 呼び出しの XML 文書を作成し、これを OneWorld に渡します。応答 XML 文書により、テーブルに存在する列のリストが渡されます。生成されたビジネス・オブジェクトは、表示されたすべての列を、ビジネス・オブジェクトの属性として持っています。

データの選択

41 ページの図 12 に示すように、この「BO プロパティー」ウィンドウでは、各テーブルの列の where 文節のプロパティーを取り込みます。このウィンドウは、それぞれのテーブルごとに表示されます。プロパティー名は、テーブルの列名です。プロパティー値は、列に使用する必要のある where 文節のパラメーターを表します。where 文節の一部を構成する列についてのみ、データが取り込まれます。値は、次の形式でなければなりません。

```
clause type=<ClauseType>;
clause_seq=<Clause Sequence>;
operator_type=<Operator Type>;
```

<Clause Type> は、WHERE、AND、または OR です。<Clause Seq> は、where 文節において列を追加する順序を表す数値です。<Operator Type> には、次のいずれかの値を指定できます。

- EQ (等しい)
- NE (等しくない)
- LT (より小)
- GT (より大)
- LE (より小か等しい)
- GE (より大か等しい)

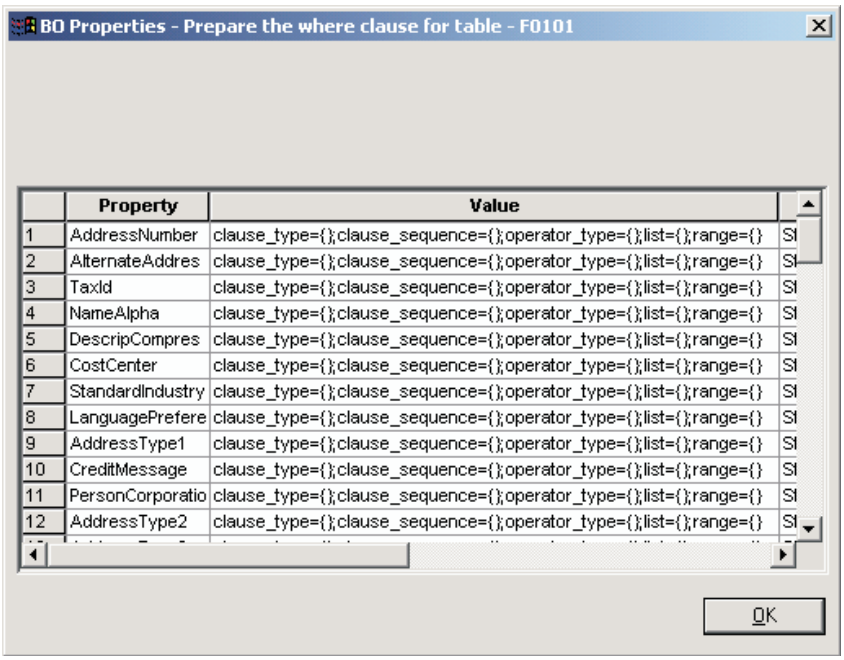

図 12. 「テーブルの where 文節の作成 (Prepare the where clause for table)」画面

データの順序付け

42 ページの図 13 に示すデータの順序付けの画面では、プロパティー名が表示され、特定の列の順序付けを昇順 (ASCD) にするか降順 (DSCD) にするかをドロップダウン・メニューから選択できます。

順序付けのタイプのほかに、アダプターでは、ASI タグ sort_order を使用して、順序付け文節の作成時に属性の順序を設定します。ODA は、このプロパティーを追加しません。このプロパティーは、ソート・プロパティーの選択時、ユーザーが手動で属性 ASI に追加する必要があります。

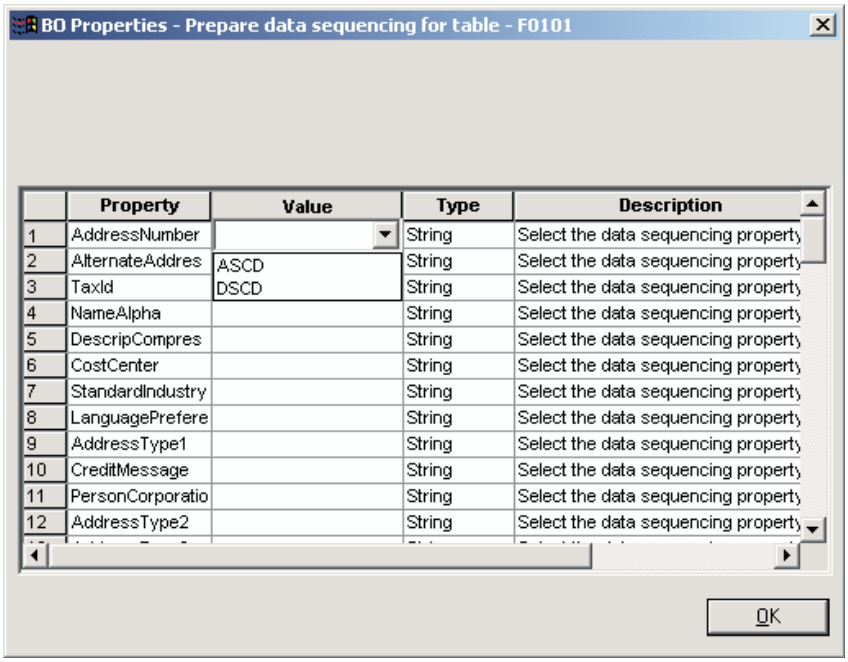

図 13. 「データの順序付けの作成 (Prepare data sequencing)」画面

別のウィンドウでビジネス・オブジェクトを開く

「*ビジネス・オブジェクト・ウィザード - ステップ 6/6 - ビジネス・オブジェクトの保管*」画面が表示され、そこに、別ファイルにビジネス・オブジェクトのコピーを保管するオプション、別ウィンドウで新規ビジネス・オブジェクトを開くオプション、および OneWorld ODA をシャットダウンするオプションが示されます。別ウィンドウで新規ビジネス・オブジェクトを開くオプションを選択したときに、Business Object Designer Express が表示するウィンドウの中で、それらのビジネス・オブジェクトの属性を変更できます。

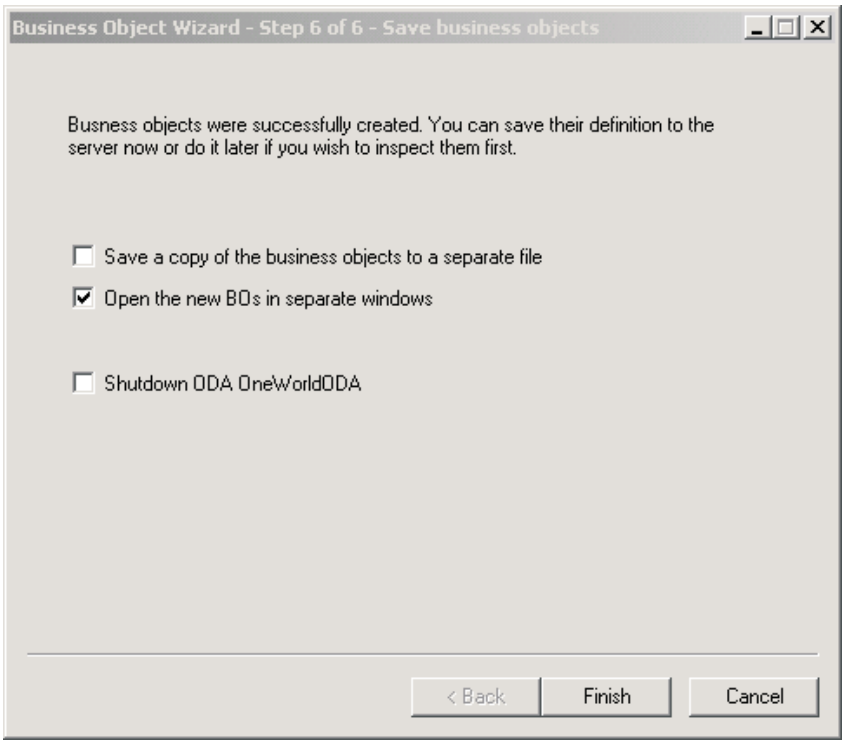

図 14. 「ビジネス・オブジェクトの保管」画面

個別のウィンドウでビジネス・オブジェクトを開くには、次の手順を実行します。

1. **「別のウィンドウで新規ビジネス・オブジェクトを開く」**を選択します。ダイアログ・ボックスが表示されます。
2. **「完了」**をクリックします。それぞれのビジネス・オブジェクトが別々のウィンドウに表示されます。そのウィンドウで、作成したばかりのビジネス・オブジェクトおよびビジネス・オブジェクト動詞の ASI 情報を表示し、設定することができます。詳細については、37 ページの『インターフェース・クラスにマップされるビジネス・オブジェクト』を参照してください。

ビジネス・オブジェクトをファイルに保管するには、次の操作を実行します (親レベルのビジネス・オブジェクト用のキーを指定しておく必要があります)。

1. **「ビジネス・オブジェクトのコピーを個別のファイルに保管する」**を選択します。ダイアログ・ボックスが表示されます。
2. 保管する新規ビジネス・オブジェクト定義のコピーを格納する場所を入力します。

Business Object Designer Express は、ファイルを指定された場所に保管します。

ODA での作業を完了後、「ODA JD Edwards OneWorld ODA をシャットダウン (Shutdown ODA JD Edwards OneWorld ODA)」チェック・ボックスを選択してから**「完了」**をクリックすることによって ODA をシャットダウンできます。

ODA の停止

Windows での ODA の停止

ODA での作業を完了した後、「**ODA JD Edwards OneWorld ODA をシャットダウン (Shutdown ODA JD Edwards OneWorld ODA)**」チェック・ボックスを選択してから「**完了**」をクリックすることによって、ODA をシャットダウンできます。

i5/OS での ODA の停止

ODA の停止は、始動するのに使用した方法により異なります。

30 ページの『i5/OS での ODA の始動』のステップ 1 または 2 で説明した方法のいずれかを使用して始動した場合は、以下を実行します。

1. CL コマンド `WRKACTJOB SBS(QWBISVR43)` を実行します。画面にサブシステムで実行中のすべてのジョブが表示されます。
2. リストをスクロールして、ODA のジョブ記述と一致するジョブ名のジョブを見つけます。ODAName JD Edwards OneWorld ODA の場合は、`QWBIJDEODA` です。
3. オプション 4 を選択して **F4** キーを押し、ENDJOB コマンドのプロンプトを表示して、OPTION パラメーターの *IMMED コマンドを指定します。
4. **Enter** キーを押します。

30 ページの『i5/OS での ODA の始動』のステップ 3 を使用して ODA を始動した場合は、`start_ODAName.sh` スクリプトを実行した **F3** キーを押します。

ビジネス・オブジェクト・ファイルのアップロード

新規に作成されたビジネス・オブジェクト定義ファイルは、作成後に統合ブローカーにアップロードする必要があります。InterChange Server Express で、ビジネス・オブジェクト定義ファイルをローカル・マシンに保管し、それらをサーバー上のリポジトリーにアップロードする必要がある場合は、「システム・インプリメンテーション・ガイド」を参照してください。

第 5 章 ビジネス・オブジェクトの理解

本章では、ビジネス・オブジェクトの構造、アダプターによるビジネス・オブジェクトの処理方法、およびそれらに関するアダプターの前提事項を説明します。

本章の内容は、次のとおりです。

メタデータの定義

JD Edwards OneWorld 対応のコネクターは、メタデータ主導型です。WebSphere Business Integration システムでは、メタデータは、OneWorld アプリケーションのオブジェクトのデータ構造を説明するアプリケーション固有の情報として定義されます。メタデータを用いて、コネクターが実行時にビジネス・オブジェクト作成に使用するビジネス・オブジェクト定義を構成します。

コネクターをインストールしても、ビジネス・オブジェクト定義を作成しなければ、コネクターを実行することができません。コネクターが処理するビジネス・オブジェクトには、統合ブローカーによって許可されている任意の名前を命名できます。命名規則については、「*コンポーネント命名ガイド*」を参照してください。

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、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。これにより、コネクターは、コードを変更しなくても新規または変更されたビジネス・オブジェクト定義を処理することができます。新規のオブジェクトは、ODA の支援がある場合でもない場合でも、Business Object Designer Express で作成されます。既存のオブジェクトを変更するには、直接 Business Object Designer Express を使用します (既存のビジネス・オブジェクトの変更に ODA を使用することはできません)。

アプリケーション固有のメタデータには、ビジネス・オブジェクトの構造およびその属性プロパティーの設定が含まれています。各ビジネス・オブジェクトの実際のデータ値は、実行時にメッセージ・オブジェクトに格納されて伝達されます。

コネクターには、サポートするビジネス・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・オブジェクト間の関係、およびデータの形式に関する前提事項があります。そのため、ビジネス・オブジェクトの構造が、対応する JD Edwards OneWorld オブジェクト用に定義された構造と正確に一致していることが重要です。異なる場合には、アダプターはビジネス・オブジェクト定義を正しく処理することができません。

ビジネス・オブジェクト構造を変更する必要がある場合は、該当するビジネス・オブジェクト構造を対応する OneWorld オブジェクトに変更してから、GenJava を実行して、ODA への入力ファイルとして使用する .jar ファイルを作成します。

ビジネス・オブジェクト定義の変更について詳しくは、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

コネクター・ビジネス・オブジェクトの構造

OneWorld オブジェクトのそれぞれに、対応するビジネス・オブジェクトがあります。

JD Edwards OneWorld アダプターのビジネス・オブジェクトには、Java API をサポートするものと、OneWorld の XMLList API をサポートするものの 2 つのタイプがあります。

ビジネス関数ビジネス・オブジェクト

Java API を使用して処理するビジネス・オブジェクトは、ビジネス・オブジェクト ASI に、このビジネス・オブジェクトと XML リスト・ビジネス・オブジェクトを識別する `type=BFN` タグを保持しています。Java API で使用するビジネス・オブジェクトの中には、2 種類のビジネス・オブジェクトがあります。1 つは `Interface` クラスにマップされているビジネス・オブジェクトであり、もう 1 つはビジネス関数クラスにマップされているビジネス・オブジェクトです。`Interface` クラスにマップされているビジネス・オブジェクトには、以下のような ASI があります。

```
type=BFN;class_name=com.JD Edwards.interop.AddressBook.JDEAddressBook
```

Java API を使用して処理するビジネス・オブジェクトは、ビジネス・オブジェクト ASI に、このビジネス・オブジェクトと XML リスト・ビジネス・オブジェクトを識別する `type=BFN` タグを保持しています。Java API で使用するビジネス・オブジェクトの中には、2 種類のビジネス・オブジェクトがあります。1 つは `Interface` クラスにマップされているビジネス・オブジェクトであり、もう 1 つはビジネス関数クラスにマップされているビジネス・オブジェクトです。`Interface` クラスにマップされているビジネス・オブジェクトには、以下のような ASI があります。

```
type=BFN;class_name=com.JD Edwards.interop.AddressBook.JDEAddressBook
```

クラス `JDEAddressBook` 内と、GenJava によって生成された .jar ファイル内に存在するすべてのビジネス関数は、メイン・ビジネス・オブジェクトの子ビジネス・オブジェクトとして表されます。トップレベル・ビジネス・オブジェクトのすべての子オブジェクトは、1 つのユーザー名およびパスワードを通じてアクセス可能なビジネス関数にマップされている必要があります。特定のビジネス関数が、異なるユーザー名またはパスワードによってアクセスされる場合、その関数は、そのユーザー名またはパスワードにアクセス可能な別個のビジネス・オブジェクト階層に属していなければなりません。ビジネス・オブジェクトの特殊なアクセス権の定義方法の詳細については、48 ページの『ビジネス関数』を参照してください。

GenJava から生成された .jar ファイル内に記述されている各データ構造クラスは、対応するビジネス・オブジェクトにマップされます。例えば、`JDEAddressBook` では、データ構造の名前は以下のようになります。

```
D0100031
D0100019
D0100032
D0100002
D0100033
```

そして、B0100031、B0100019 などに対応して作成されたビジネス関数および子ビジネス・オブジェクトにマップされます。

上記のビジネス・オブジェクトの ASI には、name= のタグが含まれます。値はデータ構造の名前です。また、bfn_name= のタグも含まれます。これらのビジネス・オブジェクトに対応するビジネス関数の名前を指定します。

属性の名前は、ビジネス・オブジェクトによって示されるメソッドの名前にマップします。例えば、データ構造が D0100033 の場合、AddressBook ビジネス・オブジェクト内の属性の名前は GetEffectiveAddress となります。この属性レベルの ASI は、ASI タグの bfn_name= を使用してメソッドの名前を示します。

属性

Data Structure クラスに記述されている各属性ごとに、ビジネス関数のビジネス・オブジェクト内で、対応するビジネス・オブジェクト属性が生成されます。属性の ASI には、OneWorld でのその属性のタイプと名前に関する情報が保持されています。例えば、属性タイプが JDEDate の場合、ASI では name=EffectiveDate;type=JDEDate と記述されています。OneWorld では、単純タイプの属性のほかにも、JDEDate および JDEMathNumeric という 2 種類のプロプラエタリー・データ・タイプをサポートします。

JDEDate

この OneWorld Java クラスでは、以下のメソッドが使用可能です。

- JDEDate() -- コンストラクター
- GetDay() -- 日付の日を戻す
- GetMonth() -- 日付の月を戻す
- GetYear() -- 日付の年を戻す
- SetDay(short) -- 日付の日を設定する
- SetMonth(short) -- 日付の月を設定する
- SetYear(short) -- 日付の年を設定する

OneWorld の日付フィールドにマップされる属性の値は、MM/DD/YYYY という形式で指定します。アダプターはこのストリング値を構文解析し、JDEDate オブジェクトで OneWorld API を呼び出して、日、月、および年の値を設定します。OneWorld からのデータをビジネス・オブジェクトに設定する必要がある場合、アダプターは get メソッドを使用して属性に値を設定します。

JDEMathNumeric

以下のメソッドが JDEMathNumeric クラスに記述されています。

- GetValue() -- 値をストリングとして戻す (例えば 12345.6789)
- SetValue() -- ストリングから値を設定する (例えば 12345.6789")

ビジネス・オブジェクト属性のタイプ

以下の表に、OneWorld でサポートされるデータ型と、それに対応する WebSphere Business Integration ビジネス・オブジェクトを示します。

表 4. ビジネス・オブジェクト属性のタイプ

OneWorld タイプ	ビジネス・オブジェクト属性のタイプ	ASI
JDEDate	Date	type=JDEDate
JDEMathNumeric	Integer	type=JDEMathNumeric
int	Integer	type=int
boolean	Boolean	type=boolean
char	String	type=char
String	String	type=String
short	Integer	type_short
float	Float	type=float
double	Double	type=double
byte	String	type=byte
long	Integer	type=long

ビジネス関数

OneWorld コネクターは、1 回のトランザクションの 1 回の doVerbFor() 呼び出しで、ビジネス関数をすべて呼び出します。そのうちの 1 つが失敗した場合、すべての関数がロールバックされます。1 回のビジネス・オブジェクト実行におけるすべてのビジネス関数は、単一のユーザーからのアクセス権が認められている必要があります。ユーザーは、アダプター用に作成され、接続プールとして維持されているユーザーでも、特定のユーザーでもかまいません。タイプが ACCESS_LEVEL である単一カーディナリティーの子ビジネス・オブジェクトを使用して、ビジネス・オブジェクトのユーザーを指定することができます。

ビジネス関数呼び出しを適切に表現するため、ビジネス関数は、データ構造変数を表す属性を含む子ビジネス・オブジェクトとして作成されます。

アダプターは独立して実行されるビジネス関数にマップされるビジネス・オブジェクトをサポートします。このようなビジネス・ オブジェクトのすべてについて、ASI には、ビジネス関数の実行に必要な情報 (データ構造のビジネス関数名など) が格納されています。上記のように、ビジネス関数ビジネス・オブジェクト ASI は以下のように表します。

```
bfn_name=getEffectiveAddress
type=BFN
name=com.JD Edwards.interop.D0100031
```

XML ビジネス・オブジェクト関数

XML ビジネス・オブジェクトは、OneWorld のテーブルにマップされます。ビジネス・オブジェクトの属性は、テーブルの列にマップされます。

コンポーネントに同じ名前の列が複数ある場合、ODA は、属性名に「_」を付加し、数値を続けることによって、固有の属性名を生成します。例えば、フィールド AddressNumber が複数回出現する場合、生成された属性名は、AddressNumber、AddressNumber_1、AddressNumber_2 のようになります。属性の最大長は、getTemplate() API 呼び出しからの戻り値に基づいて設定されます。

カスタム・ビジネス関数

イベント通知をアダプターに実装するには、以下のカスタム・ビジネス関数が必要です。

- Retrieve_WBIAEvents -- IBM イベント表からレコードを取り出すビジネス関数の名前
- Update_WBIAEvents -- IBM イベント表からのレコードを更新するビジネス関数の名前
- Delete_WBIAEvents -- IBM イベント表からレコードを削除するビジネス関数の名前
- Archive_WBIAEvents -- IBM イベント表からのレコードをアーカイブ表に保存するビジネス関数の名前
- Recover_WBIAEvents -- IN_PROGRESS イベントを検索して状況を READY_FOR_POLL に変更するビジネス関数の名前

イベント・ビジネス・オブジェクトの構造

次の表に、コネクターがサポートするイベント通知機能の詳細を示します。

表 5. イベント表の構造

カラム	説明
OBJ_KEY	イベントが作成されたビジネス・オブジェクト行を示す固有 ID。キーを作成するビジネス・オブジェクト内に複数の属性が存在する場合、値は ";" によって区切られる名前と値のペアになります。 ビジネス・オブジェクトがビジネス関数タイプの場合、オブジェクト・キーは DS0013keyattr1=123; DS0013keyattr2=124 のようになっている必要があります。retrieve 動詞の動詞 ASI が複数のビジネス関数を指定する場合、複数のキー・フィールドの設定が必要な場合があります。例えば D0013.attr1=123;D0012.attr1=345 のように、アダプターが属性名とともにデータ構造名を使用しているため、このような表記がサポートされています。
OBJ_NAME	イベントが検出された OneWorld ビジネス・オブジェクト。
OBJ_VERB	イベントの動詞。
EVT_PRIORITY	イベント優先順位。
EVT_STATUS	イベント状況。最初は READY_FOR_POLL に設定されています。
EVT_DESC	イベントに関連したコメント。
EVENT_ID	イベント行の固有 ID。
ADAPTER_ID	複数コネクター構成において接続を識別します。
EVT_TIME	イベント作成のタイム・スタンプ。

表 5. イベント表の構造 (続き)

カラム	説明
ROW_ID	OneWorld によって生成されるアーカイブ・レコード ID。
PROC_TIME	イベント処理のタイム・スタンプ

ビジネス関数のアプリケーション固有情報

ビジネス関数のアプリケーション固有情報は、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の指示をコネクターに提供します。ビジネス・オブジェクト定義を拡張または変更する場合、定義内のアプリケーション固有の情報を、コネクターが想定している構文に一致していることを確認する必要があります。

アプリケーション固有の情報は、ビジネス・オブジェクト全体に対しても、各ビジネス・オブジェクト属性に対しても指定することができます。

ビジネス・オブジェクト・レベル ASI

オブジェクト・レベル ASI は、ビジネス・オブジェクトおよびそのビジネス・オブジェクトに含まれるオブジェクトの性質に関する基礎情報を提供します。ビジネス・オブジェクト ASI は名前と値のペアです。インターフェース・オブジェクトを表すビジネス・オブジェクトは、以下の ASI 名を認識します。

- type=BFN (アダプターがビジネス関数を呼び出す場合)
- class_name=com.JD Edwards.interop.AddressBook (インターフェース・クラスの名前)

ビジネス・オブジェクトは、独立して実行されるビジネス関数にマップされます。ビジネス関数を表すビジネス・オブジェクトの場合、アダプターは以下の名前を認識します。

- type=BFN (アダプターがビジネス関数を呼び出す場合)
- bfn_name=getEffectiveAddress
- name=com.JD Edwards.interop.D0100031 (構造クラスの名前)

動詞 ASI

インターフェース・クラスにマップされるビジネス・オブジェクトの場合、動詞 ASI には、OneWorld BO Handler が呼び出すビジネス関数にマップされる一連の属性名が含まれています。アダプターは、動詞 ASI によって指定された順序でビジネス関数を呼び出します。

ビジネス関数にマップされるビジネス・オブジェクトの場合、動詞 ASI はブランクです。

属性レベルの ASI

ビジネス関数にマップされるビジネス・オブジェクトには、データ構造クラスの get<Attr>/set<Attr> メソッドの組み合わせにマップされる属性があります。コネクターは、関数呼び出し時に、このデータ構造オブジェクトを入力パラメーターとみなします。このようなすべての属性について、ASI は属性のタイプを type=<type> として保管し、属性の実際の名前を name=<name> として保管します。

アダプターが get/set の組み合わせに対して生成する属性は 1 つのみです。例えば、属性名が ID の場合、メソッドは getID() および setID() となります。そしてビジネス・オブジェクトには ID という名前の 1 つの属性があり、ASI は getter=getID();setter=setID()、type=int、name=ID となります。

表 6 に、メソッド以外の属性の ASI を示します。

表 6. メソッド以外の属性の属性レベル ASI

属性	説明
Name	ビジネス・オブジェクト・フィールド名を指定します。
Type	ビジネス・オブジェクト・フィールド・タイプを指定します。
MaxLength	デフォルトでは 255 文字です。
IsKey	false に設定します。
IsForeignKey	false に設定します。
IsRequired	false に設定します。フィールドが必須の場合、true に設定します。
AppSpecificInfo	この属性は次のようにフォーマット設定されます。 name=; type=; use_attribute_value=busobj.attrname(optional); getter=; setter=;
DefaultValue	なし

表 7 に、メソッド以外の属性の ASI を示します。

表 7. メソッド属性の属性レベルの ASI

属性	説明
Name	ビジネス・オブジェクト・フィールド名を指定します。
Type	ビジネス・オブジェクト・フィールド・タイプを指定します。
Relationship	子がコンテナー属性の場合、このフィールドは Container に設定します。
IsKey	false に設定します。
IsForeignKey	false に設定します。
IsRequired	false に設定します。
AppSpecificInfo	なし
Cardinality	1

XML ビジネス・オブジェクト関数のアプリケーション固有情報

XML ビジネス・オブジェクト関数のアプリケーション固有情報は、XML ビジネス・オブジェクト関数の処理方法に関するアプリケーション依存の指示をコネクターに提供します。XML ビジネス・オブジェクト定義を拡張または変更する場合は、定義内のアプリケーション固有情報が、コネクターが予期する構文と一致していることを確認する必要があります。

アプリケーション固有の情報は、ビジネス・オブジェクト全体に対しても、各ビジネス・オブジェクト属性に対しても指定することができます。

ビジネス・オブジェクト・レベル ASI

ビジネス・オブジェクト ASI は type=XMLList です。テーブル・タイプが TABLE_CONVERSION でない場合は、次の ASI が存在している必要があります。

- TN=<table name>: データの取り出し元のテーブル名。
- TABLE_TYPE=<table type>: テーブルのタイプ。

  テーブル・タイプは、次のいずれかになります。

  – OWTABLE
  – OWVIEW
  – FOREIGN_TABLE

動詞 ASI

このプロパティーは、XML ビジネス・オブジェクトには使用されません。

属性レベルの ASI

ビジネス・オブジェクトの属性は、テーブルの特定の列にマップされます。

以下の属性に対して、以下の ASI が設定されています。

- alias=<column name>: 取り出す列の別名。
- Name=<column name>: コンポーネント内での列名。
- type=<type>: データ型。このプロパティーは、where 文節の作成で使用します。
- operator_type=<Type of operand>。例えば、GT、LT などです。この値は、where 文節の実行時に値を比較する際に使用します。
- table=<table name>: 列が属するテーブルの名前。この ASI はテーブル変換プロセスからデータを取り出す際にのみ使用します。
- clause_type
- clause_seq
- sorting
- sorting=<ASCD, DESD>: 値が ASCD の場合、この列のデータの順序付けでは昇順を使用します。値が DESD の場合、この列データの順序付けでは降順を使用します。

ビジネス・オブジェクト・ハンドラー

汎用 OneWorld ビジネス・オブジェクト・ハンドラーは、OneWorld コンポーネントを介して公開されるコンポーネントに対して一連のビジネス関数を呼び出す可能性のあるプロセスを処理します。このハンドラーは、OneWorld の XML リスト機能を使用して、OneWorld のテーブルからデータを取り出すこともできます。新規ビジネス・オブジェクトがビジネス・オブジェクト・ハンドラーに渡されると、ビジネス・オブジェクトの ASI が読み込まれ、アダプターがビジネス・オブジェクト関数を呼び出す必要があるか、あるいは XML リスト用の XML 文書を作成する必要があるかが判別されます。ASI タグ type は、呼び出しのタイプを指定します。type=BFN の場合、アダプターは呼び出しをビジネス・オブジェクト関数呼び出し側に渡し、この呼び出し側が OneWorld Java オブジェクトのインスタンスを生成し、ビジネス関数を呼び出します。ASI が type=XMLList に指定されている場合、アダプターは CreateList API 用の XML 文書を生成します。

ビジネス・オブジェクト・プロパティーのサンプル

このセクションでは、WebSphere ビジネス・インテグレーション・ビジネス・オブジェクトの例を示します。対応する OneWorld クラスおよび Java クラスも示し、3 つの構成体にわたるマッピングが把握できるようにします。ビジネス・オブジェクトは、一致する OneWorld アプリケーション・オブジェクトから名前を継承します。

このセクションに示すサンプルは以下のとおりです。

- 『GenJava スクリプト・ファイルのサンプル』
- 54 ページの『上記の例のビジネス・オブジェクトの構造』

GenJava スクリプト・ファイルのサンプル

OneWorld では、OneWorld サーバーの一部として実行されるビジネス関数の Java ラッパーを生成する GenJava というユーティリティーを提供しています。GenJava では、iJDEScript を用いて書かれたスクリプト・ファイルが必要です。次の例では、AddressBook.cmd というスクリプト・ファイルを使用します。AddressBook.cmd では、ライブラリーを指定し、またビジネス関数のセットがモジュラー化されているインターフェースを指定しています。

GenJava を実行すると、すべてのインターフェース・クラスおよび関連するデータ構造について、Java クラス・ファイルを作成します。GenJava は生成された Java ファイルをコンパイルし、Java 文書を生成し、それらを 2 つの .jar ファイルにパッケージします。1 つは Java クラス用、もう 1 つは Java 文書用です。次のサンプルでは、AddressBookInterop.jar および AddressBookInteropDoc.jar ファイルをレンダリングします。

次のサンプルを実行するには、コマンド行から以下のように入力します。

```
GenJava /UserID JDE /Password JDE /Environment JDETest /cmd AddressBook.cmd
```

GenJava を実行するには、いくつかの方法があります。GenJava は <INSTALL>¥system¥bin32 フォルダー内に存在しています。

OneWorld 提供の「Interoperability Guide」の『Running GenJava』に関するセクションを参照してください。以下に GenJava スクリプト・ファイルのサンプルを示します。

```
# This example creates a library whose name is derived from an input parameter
# (library) if one is specified.  A default value is used otherwise.

define library JDEAddressBook

login

library JDEAddressBook
   interface AddressBook
      import B0100031
      import B0100019
      import B0100032
      import B0100002
      import B0100033
build

logout
```

このスクリプトを作成するときに、アダプター内のビジネス・オブジェクトへのインターフェース・クラスのマッピングを検討し、動詞の意図されたアクションを実行するために必要なメソッドのシーケンスとしてビジネス関数を関連付けます。例えば、ビジネス・オブジェクトが SalesOrder ビジネス・オブジェクトの場合、スクリプト・ファイル内のインターフェース SalesOrder には、アダプターを通じて SalesOrder オブジェクトでアクションを実行するために必要なすべてのビジネス関数が組み込まれている必要があります。ビジネス・オブジェクトの動詞 ASI を取り込むことによって、各動詞の一連のメソッドが実行されます。ODA を使用して、ビジネス・オブジェクト生成プロセスでこれを実行できなければなりません。ビジネス・オブジェクト生成後は、Business Object Designer Express を使用して動詞 ASI を編集することもできます。

上記の例のビジネス・オブジェクトの構造

次の図では、Business Object Designer Express における上記の例のビジネス・オブジェクトの構造を示しています。

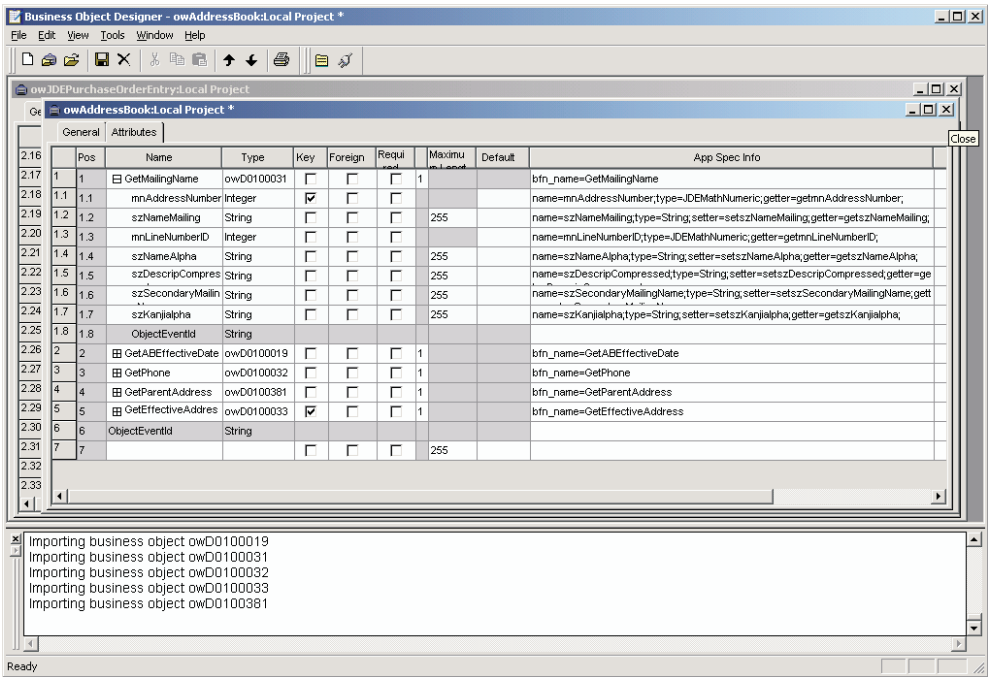

図 15. AddressBook の例のビジネス・オブジェクトの構造

このセクションでは、以降のセクションで登場する AddressBook の例のビジネス・オブジェクト構造を説明します。

AddressBook

```
Name AddressBook
ASI (type=BFN; class_name=com.JD Edwards.interop.AddressBook)
```

属性

AddressBook ビジネス・オブジェクトには、以下のオブジェクトが含まれています。

- GetMailingName (オブジェクト)
- GetABEffectiveDate (オブジェクト)
- GetPhone (オブジェクト)
- GetParentAddress (オブジェクト)
- GetEffectiveAddress (オブジェクト)

動詞 ASI

動詞 ASI は、以下の Retrieve オブジェクトおよび RetrieveDetails オブジェクトを使用します。

- Retrieve -- GetEffectiveAddress
- RetrieveDetails -- GetPhone; GetMailingName

D0100033

```
Name D0100033
ASI (type=BFN;
class_name=com.JD Edwards.interop.jdeaddressbook.D0100033;
bfn_name=GetEffectiveAddress)
```

属性

D0100033 ビジネス・オブジェクトには、以下のオブジェクトが含まれています。

- mnAddressNumber (Integer)
  (ASI) type = JDEMathNumeric; name = m_mnAddressNumber;
  use_attribute_value=;getter=getmnAddressNumber;setter=;
- jdDateBeginningEffective (Date)
  (ASI) type = JDEDate; name = m_mnAddressNumber;
  use_attribute_value=;getter=getjdDateBeginningEffective;
  setter=setjdDateBeginningEffective;

動詞 ASI

なし。

ビジネス・オブジェクトの生成

OneWorld アプリケーションは、実行時にイベントが発生するたびに、オブジェクト・レベルのデータおよびトランザクションのタイプに関する情報を含むメッセージ・オブジェクトを送信します。コネクターは、このデータを対応するビジネス・オブジェクト定義にマップして、アプリケーション固有のビジネス・オブジェクトを作成します。コネクターは、これらのビジネス・オブジェクトを処理のために統合ブローカーに送ります。また、統合ブローカーから戻されたビジネス・オブジェクトを受け取ります。そして、そのビジネス・オブジェクトを OneWorld アプリケーションに戻します。

注: OneWorld アプリケーション内でオブジェクト・モデルが変更された場合は。ODA を使用して新規の定義を作成します。統合ブローカー・リポジトリー内のビジネス・オブジェクト定義が OneWorld アプリケーションの送信したデータと正確に一致しない場合、コネクターはビジネス・オブジェクトを作成できず、トランザクションは失敗します。

Business Object Designer Express では、実行時に使用するビジネス・オブジェクト定義を作成および変更できるグラフィカル・インターフェースを提供します。詳細については、 29 ページの『第 4 章 ビジネス・オブジェクトの作成および変更』を参照してください。

第 6 章 エラー処理

この章では、Adapter for JD Edwards OneWorld のエラー処理方法を説明します。アダプターは、ロギング・メッセージおよびトレース・メッセージを生成します。この章で、これらのメッセージについて説明します。本章の内容は、次のとおりです。

エラー処理

このセクションでは、OneWorld アダプターおよび OneWorld ODA のエラー処理について説明します。

アダプター

アダプターは、OneWorld でビジネス関数を実行するときに、次の 3 種類の例外をスローします。

- 致命的
- リカバリー可能
- 拒否

致命的

FatalException クラス条件は、手操作による介入を必要とします。アダプターは致命的例外条件をキャッチし、例外のテキストを ReturnStatusDescriptor ストリングに書き込みます。戻り状況は FAIL です。

リカバリー可能

リカバリー可能エラーの場合、アダプターはビジネス関数の実行を再試行します。リカバリー可能例外が再度スローされると、アダプターは例外のテキストを ReturnStatusDescriptor ストリングに書き込みます。戻り状況は FAIL です。2 回目の試行が成功した場合、戻り状況は VALCHANGED です。

拒否

拒否例外の場合、エラーなのか警告なのかは、戻り値によって決まります。次の値が戻される可能性があります。

- Successful=0--例外が発生した場合は、この状況は戻されません。
- Warning=1--アダプターは ReturnStatusDescriptor に例外メッセージを取り込み、状況 VALCHANGED を戻します。
- Error=2--アダプターは ReturnStatusDescriptor に例外メッセージを取り込み、状況 FAIL を戻します。

XMLList ビジネス・オブジェクトの処理でエラーが発生した場合、応答 XML 文書にはエラー・コードとエラー・ストリングが格納されます。アダプターは、ReturnStatusDescriptor にエラー・コードおよびエラー・ストリングを書き込み、状況 FAIL を戻します。

ODA

OneWorld ODA は、以下のシナリオで例外をスローします。

- .jar ファイル用に指定されたパスが存在しないか、またはアクセスできない場合
- .jar ファイルが壊れているか、またはアクセスできない場合
- .jar ファイルが空の場合

ODK プロパティーによって、トレース・ファイル名およびトレース・レベルを定義します。ODK ウィザードがこれらの 2 つのプロパティーを管理します。トレース・ファイルは Crossworlds/ODA フォルダーの OneWorld フォルダーにあります。ファイルのデフォルト名は OneWorldODAtrace.txt です。エラー・メッセージおよびトレース・メッセージを格納するメッセージ・ファイルにファイルには、次のような命名規則があります。

```
BIA_<ODAAgentName>Agent.txt
```

ODAAgentName は、ODA の start ファイルに記述されている同じ名前の変数から取得した値です。ODAAgentName 変数の値を変更する場合は、メッセージ・ファイル名も変更する必要があります。エラーおよびトレース・メッセージ・ファイルは ODA メッセージ・フォルダーにあります。

トレース・ファイルおよびメッセージ・ファイルの詳細については、「*ビジネス・オブジェクト開発ガイド*」の『*例外とメッセージのトレース*』を参照してください。

ロギング

ODA のメッセージはメッセージ・ファイル BIA_*<ODAAgentName>*Agent.txt から、アダプターのメッセージは BIA_OneWorldAdapter.txt から読み込みます。

トレース

トレースはオプションのデバッグ機能であり、この機能をオンにするとコネクターの動作を密着して追跡できます。トレース・メッセージは、デフォルトでは STDOUT に書き込まれます。トレース・メッセージの構成の詳細については、26 ページの『ログ・ファイルとトレース・ファイルの使用』のコネクター構成プロパティーを参照してください。

表 8 では、コネクター・トレース・メッセージ・レベルの推奨される内容をリストしています。

表 8. トレース・メッセージの内容

レベル	説明
レベル 0	このレベルは、コネクターのバージョンを示すトレース・メッセージに使用します。このレベルでは他のトレースは実行されません。

表 8. トレース・メッセージの内容 (続き)

レベル	説明
レベル 1	このレベルは、以下の項目を実行するトレース・メッセージに使用します。 • 状況情報を提供する。 • 処理される各ビジネス・オブジェクトのキー情報を提供する。 • ポーリング・スレッドが入力キュー内で新規メッセージを検出するたびに記録する。
レベル 2	このレベルは、以下の項目を実行するトレース・メッセージに使用します。 • コネクターが処理する各オブジェクトに使用されるビジネス・オブジェクト・ハンドラーを識別する。 • ビジネス・オブジェクトが統合ブローカーにポストされるたびにログに記録する。 • 要求ビジネス・オブジェクトを受信するたびに通知する。
レベル 3	このレベルは、以下の項目を実行するトレース・メッセージに使用します。 • 処理中のサブオブジェクトを示す (該当する場合)。このメッセージは、コネクターがビジネス・オブジェクト内で外部キーを検出した場合、またはコネクターがビジネス・オブジェクト内に外部キーを設定した場合に表示されます。 • ビジネス・オブジェクト処理を示す。この例としては、ビジネス・オブジェクト間での一致の検出、子ビジネス・オブジェクトの配列内でのビジネス・オブジェクトの検索などがあります。
レベル 4	このレベルは、以下の項目を実行するトレース・メッセージに使用します。 • アプリケーション固有の情報を示す。この例としては、ビジネス・オブジェクト内のアプリケーション固有の情報フィールドを処理するメソッドによって戻される値などがあります。 • コネクターが関数を開始または終了したときにそれを示す。このメッセージによって、コネクターの処理フローをトレースすることができます。 • スレッド固有の処理を記録する。例えば、コネクターが複数のスレッドを作成した場合、メッセージがそれぞれの新規スレッドの作成をログに記録します。
レベル 5	このレベルは、以下の項目を実行するトレース・メッセージに使用します。 • コネクター初期化を示す。このタイプのメッセージには、例えば、ブローカーから取り出した各 Connector Configurator Express プロパティーの値が含まれます。 • コネクターが実行中に作成した各スレッドの状況の詳細を示す。 • アプリケーション内で実行されたステートメントを表す。コネクター・ログ・ファイルには、ターゲット・アプリケーションで実行されたすべてのステートメントおよび置換された変数の値 (該当する場合) が記述されます。 • ビジネス・オブジェクトのダンプを記録する。コネクターは、処理を開始する前には、コネクターがコラボレーションから受け取ったオブジェクトを示すビジネス・オブジェクトのテキスト表現を出力し、オブジェクトの処理終了後には、コネクターがコラボレーションに戻すオブジェクトを示すビジネス・オブジェクトのテキスト表現を出力します。

付録 A. コネクターの標準構成プロパティー

この付録では、WebSphere Business Integration Server Express アダプターのコネクター・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。説明は、InterChange Server Express が対象となります。

このコネクター固有のプロパティーの詳細については、本書の該当するセクションを参照してください。

新規プロパティー

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。

- AdapterHelpName
- ControllerEventSequencing
- jms.ListenerConcurrency
- jms.TransportOptimized
- TivoliTransactionMonitorPerformance

標準コネクター・プロパティーの概要

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。フレームワークが使用します。
- アプリケーション固有またはコネクター固有の構成プロパティー。エージェントが使用します。

これらのプロパティーによって、アダプターのフレームワークとエージェントの実行時の振る舞いが決まります。

このセクションでは、Connector Configurator Express の始動方法について説明し、すべてのプロパティーに共通する特性について説明します。コネクター固有の構成プロパティーについては、該当するアダプターのユーザーズ・ガイドを参照してください。

Connector Configurator Express の始動

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

構成プロパティー値の概要

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。

1. デフォルト
2. InterChange Server Express 統合ブローカー用のリポジトリー

3. ローカル構成ファイル
4. コマンド行

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。 STRING プロパティー・タイプには長さの制限はありません。 INTEGER タイプの長さは、アダプターが稼働しているサーバーによって決められます。

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。

プロパティーの更新特性、つまり、コネクターのプロパティーへの変更を有効にする方法とタイミングは、プロパティーの性質によって異なります。

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、新しい値が即時に有効になります。ただし、コネクターがスタンドアロン・モードである (System Manager から独立している) 場合です。
- **エージェント再始動 (InterChange Server Express のみ)**
  コネクター・エージェントを停止して再始動するまで、新しい値は有効になりません。
- **コンポーネント再始動**
  System Manager でコネクターを停止してから再始動しなければ、新しい値が有効になりません。エージェントまたはサーバー・プロセスを停止して再始動する必要はありません。
- **システム再始動**
  コネクター・エージェントおよびサーバーを停止して再始動するまで、新しい値は有効になりません。

特定のプロパティーの更新方法を確認するには、「Connector Configurator Express」ウィンドウ内の「**更新メソッド**」列を参照するか、63 ページの表 9 の「更新メソッド」列を参照してください。

標準プロパティーが存在できるロケーションは 3 つあります。複数のロケーションにあるプロパティーもあります。

- **ReposController**
  このプロパティーはコネクター・コントローラー内にあり、そこでのみ有効になります。エージェント側の値を変更しても、コントローラーには影響しません。
- **ReposAgent**
  このプロパティーはエージェント内にあり、そこでのみ有効になります。このプロパティーに応じて、ローカル構成でこの値を指定変更できます。
- **LocalConfig**
  このプロパティーは、コネクターの構成ファイル内にあり、構成ファイルを通じてのみ機能することができます。コントローラーはこのプロパティーの値を変更できません。また、システムを再配置してコントローラーを明示的に更新しない限り、構成ファイルに加えられた変更を認識しません。

標準プロパティーの早見表

表 9 は、標準コネクター構成プロパティーの早見表です。すべてのコネクターがこれらのプロパティーすべてを必要とするわけではありません。プロパティーの設定はそれぞれ異なる可能性があります。

各プロパティーの詳細については、表の次のセクションを参照してください。

注: 表 9 の注の欄で「RepositoryDirectory が REMOTE に設定されている」という句は、ブローカーが InterChange Server Express であることを示します。

表 9. 標準構成プロパティーの要約

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
AdapterHelpName	有効な <Regional Setting> ディレクトリーを含む <ProductDir>¥bin¥Data¥App¥Help 内の有効なサブディレクトリーの 1 つ	テンプレート名 (有効な場合) またはブランク・フィールド	コンポーネント再始動	サポートされる地域設定。 chs_chn、cht_twn、deu_deu、esn_esp、fra_fra、ita_ita、jpn_jpn、kor_kor、ptb_bra、および enu_usa (デフォルト) を含みます。
AdminInQueue	有効な JMS キュー名	*<CONNECTORNAME>* /ADMININQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
AdminOutQueue	有効な JMS キュー名	*<CONNECTORNAME>* /ADMINOUTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
AgentConnections	1 から 4	1	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が MQ または IDL で、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
AgentTraceLevel	0 から 5	0	ICS では動的、その他の場合はコンポーネント再始動	
ApplicationName	アプリケーション名	コネクター・アプリケーション名に指定された値	コンポーネント再始動	
BrokerType	ICS、	ICS	コンポーネント再始動	
CharacterEncoding	サポートされる任意のコード。次のリストはその一部です。 ascii7、ascii8、SJIS、Cp949、GBK、Big5、Cp297、Cp273、Cp280、Cp284、Cp037、Cp437	ascii7	コンポーネント再始動	このプロパティーは、C++ コネクターに対してのみ有効です。

表 9. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
CommonEventInfrastructure	true または false	false	コンポーネント再始動	
CommonEventInfrastructureURL	URL ストリング。例えば、corbaloc:iiop:host:2809.	デフォルト値はありません。	コンポーネント再始動	このプロパティーは、CommonEvent Infrastructure の値が true である場合のみ有効です。
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント再始動	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
ContainerManagedEvents	ブランクまたは JMS	ブランク	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
ControllerEventSequencing	true または false	true	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
ControllerStoreAndForwardMode	true または false	true	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
ControllerTraceLevel	0 から 5	0	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
DeliveryQueue	任意の有効な JMS キュー名	*<CONNECTORNAME>* /DELIVERYQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
DeliveryTransport	IDL または JMS	RepositoryDirectory の値が <REMOTE> である場合は IDL。それ以外の場合は JMS。	コンポーネント再始動	RepositoryDirectory の値が <REMOTE> ではない場合、このプロパティーで有効な値は JMS のみです。

表 9. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
DuplicateEventElimination	true または false	false	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
EnableOidForFlowMonitoring	true または false	false	コンポーネント再始動	このプロパティーは、BrokerType の値が ICS である場合のみ有効です。
FaultQueue	任意の有効なキュー名。	*<CONNECTORNAME>*/FAULTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory、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	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
jms.ListenerConcurrency	1 から 32767	1	コンポーネント再始動	このプロパティーは、jms.TransportOptimized の値が true の場合にのみ有効です。
jms.MessageBrokerName	jms.FactoryClassName の値が IBM である場合は、crossworlds.queue.manager を使用します。	crossworlds.queue.manager	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
jms.Password	任意の有効なパスワード		コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
jms.TransportOptimized	true または false	false	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS で、BrokerType の値が ICS である場合のみ有効です。
jms.UserName	任意の有効な名前		コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICSである場合のみ有効です。

表 9. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
ListenerConcurrency	1 から 100	1	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が MQ である場合のみ有効です。
Locale	これは、サポートされるロケールの一部です。en_US 、ja_JP 、ko_KR 、zh_CN 、zh_TW 、fr_FR 、de_DE 、it_IT 、 es_ES 、pt_BR	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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
MaxEventCapacity	1 から 2147483647	2147483647	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
MessageFileName	有効なファイル名	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	このプロパティーは、DuplicateEventElimination の値が true で、ContainerManagedEvents に値がない場合のみ有効です。
OADAutoRestartAgent	true または false	false	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。

表 9. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
OADMaxNumRetry	正整数	1000	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
OADRetryTimeInterval	正整数 (単位: 分)	10	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
PollEndTime	HH = 0 から 23 MM = 0 から 59	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒)	10000	ブローカーが ICS の場合は動的。 それ以外の場合はコンポーネント再始動	
PollQuantity	1 から 500	1	エージェント再始動	このプロパティーは、ContainerManagedEvents の値が JMS である場合のみ有効です。
PollStartTime	HH = 0 から 23 MM = 0 から 59	HH:MM	コンポーネント再始動	
RepositoryDirectory	ブローカーが ICS である場合は <REMOTE>。 それ以外の場合は、任意の有効なローカル・ディレクトリー。	ICS の場合、値は <REMOTE> に設定されます。	エージェント再始動	
RequestQueue	有効な JMS キュー名	*<CONNECTORNAME>*/REQUESTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
ResponseQueue	有効な JMS キュー名	*<CONNECTORNAME>*/RESPONSEQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
RestartRetryCount	0 から 99	3	ICS の場合は動的、その他の場合はコンポーネント再始動	
RestartRetryInterval	1 から 2147483647 の値 (単位: 分)	1	ICS の場合は動的、その他の場合はコンポーネント再始動	

表 9. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
RHF2MessageDomain	mrmまたは xml	mrm	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS で、WireFormat の値が CwXML である場合のみ有効です。
SourceQueue	任意の有効な WebSphere MQ キュー名	*<CONNECTORNAME>*/SOURCEQUEUE	エージェント再始動	このプロパティーは、ContainerManagedEvents の値が JMS である場合のみ有効です。
SynchronousRequest Queue	任意の有効なキュー名。	*<CONNECTORNAME>*/SYNCHRONOUSREQUEST QUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
SynchronousRequest Timeout	0 から任意の数 (単位: ミリ秒)	0	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
SynchronousResponse Queue	任意の有効なキュー名	*<CONNECTORNAME>*/SYNCHRONOUSRESPONSE QUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。
TivoliMonitorTransaction Performance	true または false	false	コンポーネント再始動	
WireFormat	CwXMLまたは CwBO	CwXML	エージェント再始動	RepositoryDirectory の値が <REMOTE> に設定されていない場合、このプロパティーの値は CwXML である必要があります。RepositoryDirectory の値が <REMOTE> に設定されている場合、値は CwBO である必要があります。
WsifSynchronousRequest Timeout	0 から任意の数 (単位: ミリ秒)	0	コンポーネント再始動	BrokerType の値が ICS である場合、このプロパティーは無効です。
XMLNameSpaceFormat	shortまたは long	short	エージェント再始動	BrokerType の値が ICS である場合、このプロパティーは無効です。

標準プロパティー

このセクションでは、標準コネクター構成プロパティーについて説明します。

AdapterHelpName

AdapterHelpName プロパティーは、コネクター固有の全般ヘルプ・ファイルがあるディレクトリーの名前です。ディレクトリーは *<ProductDir>*¥bin¥Data¥App¥Help に

配置され、少なくとも言語ディレクトリー enu_usa を含んでいる必要があります。これには、ロケールに従ってその他のディレクトリーが含まれることがあります。

テンプレート名が有効である場合は、それがデフォルト値になります。それ以外の場合は、ブランクになります。

AdminInQueue

AdminInQueue プロパティーは、統合ブローカーが管理メッセージをコネクターに送信するときに使用するキューを指定します。

デフォルト値は *<CONNECTORNAME>*/ADMININQUEUE です。

AdminOutQueue

AdminOutQueue プロパティーは、コネクターが統合ブローカーへ管理メッセージを送信するときに使用するキューを指定します。

デフォルト値は *<CONNECTORNAME>*/ADMINOUTQUEUE です。

AgentConnections

AgentConnections プロパティーは、ORB (オブジェクト・リクエスト・ブローカー) の初期化時に開かれる ORB 接続の数を制御します。

このプロパティーのデフォルト値は 1 です。

AgentTraceLevel

AgentTraceLevel プロパティーは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を設定します。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。

デフォルト値は 0 です。

ApplicationName

ApplicationName プロパティーは、コネクター・アプリケーションの名前を一意的に識別します。この名前は、システム管理者が統合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。

デフォルトはコネクターの名前です。

BrokerType

BrokerType プロパティーは、使用する統合ブローカー・タイプを指定します。値は ICS です。

CharacterEncoding

CharacterEncoding プロパティーは、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++ コネクターでは、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、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。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 (*<ProductDir>*) にある ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。詳細については、本書の付録『Connector Configurator Express』を参照してください。

ConcurrentEventTriggeredFlows

ConcurrentEventTriggeredFlows プロパティーは、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するビジネス・オブジェクトの数に設定します。例えば、このプロパティーの値を 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。

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、次のプロパティーを構成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する必要があります。
- 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処理するよう構成されている必要があります。

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。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 1 です。

ContainerManagedEvents

ContainerManagedEvents 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、一つの JMS トランザクションとして宛先キューに配置されます。

このプロパティーを 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 = /SOURCEQUEUE

また、MimeType、およびDHClass (データ・ハンドラー・クラス)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。また、DataHandlerConfigMOName (オプションのメタオブジェクト名) を追加することもできます。これらのプロパティーの値を設定するには、Connector Configurator Expressの「**データ・ハンドラー**」タブを使用します。

これらのプロパティーはアダプター固有ですが、次にいくつかの値を例として示します。

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

「**データ・ハンドラー**」タブのこれらの値のフィールドは、ContainerManagedEventsプロパティーの値を 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。

ContainerManagedEvents プロパティーは、DeliveryTransport プロパティーの値が JMS に設定されている場合のみ有効です。

デフォルト値はありません。

ControllerEventSequencing

ControllerEventSequencing プロパティーを使用すると、コネクター・コントローラーでイベントの順序付けを使用できます。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType は ICS) のみ有効です。

デフォルト値は true です。

ControllerStoreAndForwardMode

ControllerStoreAndForwardMode プロパティーは、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 InterChange Server Express (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。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType プロパティーの値は ICS) のみ有効です。

デフォルト値は true です。

ControllerTraceLevel

ControllerTraceLevel プロパティーは、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を設定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 0 です。

DeliveryQueue

DeliveryQueue プロパティーは、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を定義します。

このプロパティーは、DeliveryTransport プロパティーの値が JMS に設定されている場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/DELIVERYQUEUE です。

DeliveryTransport

DeliveryTransport プロパティーは、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。Java Messaging Service の場合、値は JMS です。

- RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合、DeliveryTransport プロパティーの値は IDL または JMS で、デフォルトは IDL です。
- RepositoryDirectory プロパティーの値がローカル・ディレクトリーの場合は、指定可能な値は JMS のみです。

RepositoryDirectory プロパティーの値が IDL である場合、コネクターは、CORBA IIOP を使用してサービス呼び出し要求と管理メッセージを送信します。

デフォルト値は JMS です。

JMS

JMS トランスポート機構は、Java Messaging Service (JMS) を使用しての、コネクターとクライアント・コネクター・フレームワークとの間の通信を可能にします。

JMS をデリバリー・トランスポートとして選択すると、jms.MessageBrokerName、jms.FactoryClassName、jms.Password、jms.UserName などの追加の JMS プロパテ

ィーが Connector Configurator Express にリストされます。jms.MessageBrokerName プロパティーと jms.FactoryClassName プロパティーは、このトランスポートの必須プロパティーです。

InterChange Server Espress (ICS) が統合ブローカーである場合に以下の環境でコネクターに 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があります。

この環境では、WebSphere MQ クライアント内でメモリーが使用されるため、(サーバー・サイドの) コネクター・コントローラーと (クライアント・サイドの) 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 768MB 未満である場合は、次の変数とプロパティーを設定してください。

- 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 (<*ProductDir*>) の下の ¥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、「*WebSphere Business Integration Server Express インストール・ガイド (Windows 版または i5/OS 版)*」を参照してください。

DuplicateEventElimination

このプロパティーの値が true の場合、JMS 対応コネクターでは重複イベントがデリバリー・キューへデリバリーされないようにすることができます。この機能を使用するには、コネクターの開発時に、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。

注: このプロパティーの値が true である場合は、保証付きイベント・デリバリーを提供するため MonitorQueue プロパティーを使用可能にする必要があります。

デフォルト値は false です。

EnableOidForFlowMonitoring

このプロパティーの値が true である場合、アダプター・ランタイムは、着信 ObjectEventID をフロー・モニター用の外部キーとしてマークします。

このプロパティーは、BrokerType プロパティーが ICS に設定されている場合のみ有効です。

デフォルト値は false です。

FaultQueue

コネクターでメッセージを処理中にエラーが発生すると、コネクターは、FaultQueue プロパティーで指定されているキューにそのメッセージ (および状況標識と問題説明) を移動します。

デフォルト値は *<CONNECTORNAME>*/FAULTQUEUE です。

jms.FactoryClassName

jms.FactoryClassName プロパティーは、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。このプロパティーは、DeliveryTransport プロパティーの値が JMS である場合に設定する必要があります。

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

jms.ListenerConcurrency

jms.ListenerConcurrency プロパティーは、JMS コントローラーの並行リスナーの数を指定します。これは、コントローラー内で並行して複数のメッセージを取り出して処理するスレッドの数を指定します。

このプロパティーは、jms.OptimizedTransport プロパティーの値が true である場合のみ有効です。

デフォルト値は 1 です。

jms.MessageBrokerName

jms.MessageBrokerName は、JMS プロバイダーのために使用するブローカー名を指定します。JMS をデリバリー・トランスポート機構として (DeliveryTransport プロパティーで) 指定する場合は、このコネクター・プロパティーを設定する必要があります。

リモート・メッセージ・ブローカーに接続すると、このプロパティーは次の値を要求します。
QueueMgrName:*Channel*:*HostName*:*PortNumber*
ここで、以下のように説明されます。
QueueMgrName は、キュー・マネージャー名です。
Channel は、クライアントが使用するチャネルです。
HostName は、キュー・マネージャーの配置先のマシン名です。
PortNumber は、キュー・マネージャーが listen に使用するポートの番号です。

例えば、次のように指定します。

```
jms.MessageBrokerName = WBIMB.Queue.Manager:CHANNEL1:RemoteMachine:1456
```

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

jms.NumConcurrentRequests

jms.NumConcurrentRequests プロパティーは、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するまで処理されません。

デフォルト値は 10 です。

jms.Password

jms.Password プロパティーは、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。

デフォルト値はありません。

jms.TransportOptimized

jms.TransportOptimized プロパティーは、WIP (処理中の作業) が最適化されているかどうかを判別します。WIP を最適化するには、WebSphere MQ プロバイダーが必要です。最適化された WIP を操作するには、メッセージング・プロバイダーが次のことを実行できなければなりません。

1. メッセージをキューから取り出さずに読み取る
2. 受信側のメモリー空間へメッセージ全体を転送することなく、特定の ID を使用してメッセージを削除する
3. 特定の ID を使用してメッセージを読み取る (リカバリーのために必要)
4. 読み取られなかったイベントが現れるポイントを追跡する

JMS API は、上記の条件 2 と 4 を満たさないため、最適化された WIP には使用できませんが、MQ Java API は 4 つの条件をすべて満たすので、最適化された WIP に必要です。

このプロパティーは、DeliveryTransport の値が JMS で、BrokerType の値が ICS である場合のみ有効です。

デフォルト値は false です。

jms.UserName

jms.UserName プロパティーは、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。

デフォルト値はありません。

JvmMaxHeapSize

JvmMaxHeapSize プロパティーは、エージェントの最大ヒープ・サイズ (メガバイト単位) を指定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 128MB です。

JvmMaxNativeStackSize

JvmMaxNativeStackSize プロパティーは、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) を指定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 128KB です。

JvmMinHeapSize

JvmMinHeapSize プロパティーは、エージェントの最小ヒープ・サイズ (メガバイト単位) を指定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 1MB です。

ListenerConcurrency

ListenerConcurrency 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 WebSphere 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。

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でのみ有効です。DeliveryTransport プロパティーの値は `MQ` でなければなりません。

デフォルト値は 1 です。

Locale

Locale プロパティーは、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 は、2 文字の言語コード (小文字を使用) です。
TT は 2 文字の国または地域コード (大文字) です。
codeset は、関連文字コード・セットの名前です (オプションの場合もあります)。

デフォルトでは、サポートされるロケールの一部のみがリストされます。リストに、サポートされる他の値を追加するには、*<ProductDir>*¥bin ディレクトリーにある ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。詳細については、本書の Connector Configurator Express の付録を参照してください。

コネクターが国際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、そのコネクターのユーザーズ・ガイドを参照してください。

デフォルト値は en_US です。

LogAtInterchangeEnd

LogAtInterchangeEnd プロパティーは、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。

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した場合には InterchangeSystem.cfg ファイルの MESSAGE_RECIPIENT に指定された値を宛先とする電子メール・メッセージが生成されます。例えば、LogAtInterChangeEnd の値が true の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。

このプロパティーは、RespositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値は ICS) のみ有効です。

デフォルト値は false です。

MaxEventCapacity

MaxEventCapacity プロパティーは、コントローラー・バッファー内のイベントの最大数を指定します。このプロパティーは、フロー制御機能で使用されます。

このプロパティーは、RespositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値は ICS) のみ有効です。

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

MessageFileName プロパティーは、コネクター・メッセージ・ファイルの名前を指定します。メッセージ・ファイルの標準位置は、製品ディレクトリーの ¥connectors¥messages です。メッセージ・ファイルが標準位置に格納されていない場合は、メッセージ・ファイル名を絶対パスで指定します。

コネクター・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: コネクターに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。

デフォルト値は InterchangeSystem.txt です。

MonitorQueue

MonitorQueue プロパティーは、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を指定します。

このプロパティーは、DeliveryTransport プロパティーの値が JMS で、DuplicateEventElimination の値が true である場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/MONITORQUEUE です。

OADAutoRestartAgent

OADAutoRestartAgent プロパティーは、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、WebSphere 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。WebSphere MQ により起動される OAD 機能の構成方法については、「*WebSphere Business Integration Server Express インストール・ガイド (Windows 版)*」または「*WebSphere Business Integration Server Express インストール・ガイド (i5/OS 版)*」を参照してください。

このプロパティーは、RespositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値は ICS) のみ有効です。

デフォルト値は false です。

OADMaxNumRetry

OADMaxNumRetry プロパティーは、異常シャットダウンの後で WebSphere MQ により起動される Object Activation Daemon (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。

このプロパティーは、RespositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値は ICS) のみ有効です。

デフォルト値は 1000 です。

OADRetryTimeInterval

OADRetryTimeInterval プロパティーは、WebSphere MQ により起動される Object Activation Daemon (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、コネクター・コントローラーは OAD に対してコネクター・エージェントを再始動するように要求します。OAD は OADMaxNumRetry プロパティーに指定した回数だけこの再試行処理を繰り返します。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。

このプロパティーは、RespositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値は ICS) のみ有効です。

デフォルト値は 10 です。

PollEndTime

PollEndTime プロパティーは、イベント・キューのポーリングを停止する時刻を指定します。形式は *HH:MM* です。ここで、*HH* は 0 から 23 時、*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。デフォルト値は値を含まない HH:MM であるため、この値は変更する必要があります。

アダプター・ランタイムが次のことを検出します。

- PollStartTime が設定され、PollEndTime が設定されていない、または、
- PollEndTime が設定され、PollStartTime が設定されていない

これは、PollFrequency プロパティーに対して構成された値を使用してポーリングします。

PollFrequency

PollFrequency プロパティーは、あるポーリング・アクションの終了から次のポーリング・アクションの開始までの時間をミリ秒単位で指定します。これはポーリング・アクション間の間隔ではありません。この論理を次に説明します。

- ポーリングし、PollQuantity プロパティーの値により指定される数のオブジェクトを取得します。
- これらのオブジェクトを処理します。一部のコネクターでは、これは個別のスレッドで部分的に実行されます。これにより、次のポーリング・アクションまで処理が非同期に実行されます。
- PollFrequency プロパティーで指定された間隔にわたって遅延します。
- このサイクルを繰り返します。

このプロパティーでは、次の値が有効です。

- ポーリング・アクション間のミリ秒数 (正整数)。
- 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。
- 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。

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。このようなコネクターが存在する場合には、アダプターのインストールと構成に関する章で制約事項が説明されています。

PollQuantity

PollQuantity プロパティーは、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。

このプロパティーは、DeliveryTransport プロパティーの値が JMS で、ContainerManagedEvents プロパティーに値がある場合のみ有効です。

電子メール・メッセージもイベントと見なされます。コネクターは、電子メールに関するポーリングを受けたときには次のように動作します。

- コネクターは、1 回目のポーリングを受けると、メッセージの本文を検出し、それを添付として読み取ります。本文の MIME タイプにはデータ・ハンドラーが指定されていないので、コネクターはメッセージを無視します。
- コネクターは BO の最初の添付を処理します。この添付の MIME タイプには対応するデータ・ハンドラーがあるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
- 2 回目のポーリングを受けると、コネクターは BO の 2 番目の添付を処理します。この添付の MIME タイプには対応するデータ・ハンドラーがあるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
- これが受け入れられると、BO の 3 番目の添付が送信されます。

PollStartTime

PollStartTime プロパティーは、イベント・キューのポーリングを開始する時刻を指定します。形式は *HH:MM* です。ここで、*HH* は 0 から 23 時、*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。デフォルト値は値を含まない HH:MM であるため、この値は変更する必要があります。

アダプター・ランタイムが次のことを検出します。

- PollStartTime が設定され、PollEndTime が設定されていない、または、
- PollEndTime が設定され、PollStartTime が設定されていない

これは、PollFrequency プロパティーに対して構成された値を使用してポーリングします。

RepositoryDirectory

RepositoryDirectory プロパティーは、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 Express リポジトリーからこの情報を取得するためです。

RequestQueue

RequestQueue プロパティーは、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を指定します。

このプロパティーは、DeliveryTransport プロパティーの値が JMS である場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/REQUESTQUEUE です。

ResponseQueue

ResponseQueue プロパティーは、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 InterChange Server Express (ICS) である場合、統合ブローカーは、要求を送信した後、JMS 応答キューで応答メッセージを待機します。

このプロパティーは、DeliveryTransport プロパティーの値が JMS である場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/RESPONSEQUEUE です。

RestartRetryCount

RestartRetryCount プロパティーは、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接続の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クライアント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

RestartRetryInterval プロパティーは、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リンクの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クライアント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。

このプロパティーに使用できる値は、1 から 2147483647 までです。

デフォルト値は 1 です。

RHF2MessageDomain

RHF2MessageDomain プロパティーにより、JMS ヘッダーのドメイン名フィールドの値を構成できます。JMS トランスポートを介してデータを WebSphere メッセージ・ブローカーに送信するときに、アダプター・フレームワークにより JMS ヘッダー情報、ドメイン名、および固定値 mrm が書き込まれます。この構成可能なドメイン名により、WebSphere Message Broker がメッセージ・データを処理する方法を追跡できます。

これはヘッダーの例です。

```
<mcd><Msd>mrm</Msd><Set>3</Set><Type>
Retek_POPhyDesc</Type><Fmt>CwXML</Fmt></mcd>
```

BrokerType の値が ICS である場合、このプロパティーは無効です。 また、このプロパティーは、DeliveryTransport プロパティーの値が JMS で、WireFormat プロパティーの値が CwXML である場合のみ有効です。

指定可能な値は、mrm および xml です。デフォルト値は mrm です。

SourceQueue

SourceQueue プロパティーは、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、70 ページの『ContainerManagedEvents』を参照してください。

このプロパティーは、DeliveryTransport の値が JMS で、ContainerManagedEvents の値が指定されている場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/SOURCEQUEUE です。

SynchronousRequestQueue

SynchronousRequestQueue プロパティーは、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、同期要求キューにメッセージを送信し、同期応答キューでブローカーからの応答を待ちます。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。

このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/SYNCHRONOUSREQUESTQUEUE です。

SynchronousRequestTimeout

SynchronousRequestTimeout プロパティーは、コネクターが同期要求への応答を待機する時間をミリ秒単位で指定します。指定された時間内に応答を受信できなかった場合、コネクターは元の同期要求メッセージ (およびエラー・メッセージ) を障害キューに移動します。

このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。

デフォルト値は 0 です。

SynchronousResponseQueue

SynchronousResponseQueue プロパティーは、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。

このプロパティーは、DeliveryTransport の値が JMS である場合のみ有効です。

デフォルトは *<CONNECTORNAME>*/SYNCHRONOUSRESPONSEQUEUE です。

TivoliMonitorTransactionPerformance

TivoliMonitorTransactionPerformance プロパティーは、IBM Tivoli Monitoring for Transaction Performance (ITMTP) が実行時に起動されるかどうかを指定します。

デフォルト値は false です。

WireFormat

WireFormat プロパティーは、トランスポートのメッセージ・フォーマットを指定します。

- RepositoryDirectory プロパティーの値がローカル・ディレクトリーの場合は、値は CwXML です。
- RepositoryDirectory プロパティーの値がリモート・ディレクトリーの場合、値は CwBO です。

付録 B. Connector Configurator Express

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

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

- コネクターを構成するためのコネクター固有のプロパティー・テンプレートを作成する
- 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

この付録では、次のトピックについて説明します。

Connector Configurator Express の概要

Connector Configurator Express では、InterChange Server Express 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。

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

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成する。
- **コネクター構成ファイル**を作成する。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。
  コネクター・テンプレート内のプロパティーに設定されているデフォルト値の変更が必要となります。また、サポートされるビジネス・オブジェクト定義を指定し、ICS によって、コラボレーションで使用するマップを指定する必要があります。さらに、メッセージング、ロギング、およびトレースのパラメーターを指定し、必要に応じてデータ・ハンドラー・パラメーターを指定する必要があります。

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持つプロパティー) と、コネクター固有のプロパティー (特定のアプリケーションまたはテクノロジーのためにコネクターで必要なプロパティー) とが含まれます。

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

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

ただし**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクター固有プロパティーのテンプレートを作成します。システム内で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、87 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。

Connector Configurator Express の始動

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

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

スタンドアロン・モードでの Configurator の実行

System Manager を実行せずに Connector Configurator Express を実行し、コネクター構成ファイルを編集することができます。

これを行うには、以下のステップを実行します。

- 「**スタート**」>「**すべてのプログラム**」から「**IBM WebSphere Business Integration Express**」>「**Toolset Express**」>「**開発**」>「**Connector Configurator Express**」をクリックします。
- 「**ファイル**」>「**新規**」>「**コネクター構成**」を選択します。
- 「**システム接続 Integration Broker**」の隣のプルダウン・メニューをクリックして、ICS を選択します。

Connector Configurator Express を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存する方法が便利です (93 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 Express を実行できます。

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

1. System Manager を開きます。
2. 「System Manager」ウィンドウで、「**統合コンポーネント・ライブラリー**」アイコンを展開し、「**コネクター**」を強調表示します。
3. System Manager メニュー・バーから、「**ツール**」>「**Connector Configurator Express**」をクリックします。「Connector Configurator Express」ウィンドウが開き、「**新規コネクター**」ダイアログ・ボックスが表示されます。

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て、ICS を選択します。

既存の構成ファイルを編集するには、以下のステップを実行します。

- 「System Manager」ウィンドウの「コネクター」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 Express が開き、この構成ファイルの統合ブローカー・タイプおよびファイル名が上部に表示されます。
- Connector Configurator Express で「**ファイル**」>「**開く**」を選択します。プロジェクトまたはプロジェクトが保管されているディレクトリーからコネクター構成ファイルを選択します。
- 「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、または既存のコネクター定義をテンプレートとして使用します。

- テンプレートの新規作成については、87 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。既存のテンプレートは ¥WebSphereAdapters¥bin¥Data¥App ディレクトリーにあります。

新規テンプレートの作成

このセクションでは、テンプレートでプロパティーを作成し、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。

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

1. 「**ファイル**」>「**新規**」>「**コネクター固有プロパティー・テンプレート**」をクリックします。
2. 「**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。
   - 「**新規テンプレート名を入力してください**」の下の「**名前**」フィールドに、新規テンプレートの名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
   - テンプレートに含まれているコネクタ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。

3. 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。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。
   - 既存のテンプレートを変更する場合には、「**変更する既存のテンプレートを選択してください: 検索テンプレート**」の下の「**テンプレート名**」テーブルのリストから、テンプレート名を選択します。
   - このテーブルには、現在使用可能なすべてのテンプレートの名前が表示されます。テンプレートを検索することもできます。

一般特性の指定

「**次へ**」をクリックしてテンプレートを選択すると、「**プロパティー: 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
  プロパティー・サブタイプ
  更新されたメソッド
  説明
- **フラグ**
  標準のフラグ
- **カスタム・フラグ**
  フラグ

「**プロパティー・サブタイプ**」は、「**プロパティー・タイプ**」が String である場合に選択できます。これは、構成ファイルの保管時に構文検査を提供するオプションの値です。デフォルトはブランクのスペースであり、プロパティーのサブタイプが指定されていないことを意味します。

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。値の編集も可能です。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**」のフィールドに値を入力します。

新規プロパティー値を作成するには、以下のステップを実行します。

1. 「値」列見出しの左側にある正方形を右マウス・ボタンでクリックします。
2. ポップアップ・メニューから、「**追加**」を選択して「プロパティー値」ダイアログ・ボックスを表示します。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。

3. 新規プロパティー値を入力し、「OK」をクリックします。右側の「**値**」パネルに値が表示されます。

「**値**」パネルには、3 つの列からなるテーブルが表示されます。

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

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

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

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

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

依存関係の設定

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

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

1. 「**使用可能なプロパティー**」表示で、依存プロパティーとして指定するプロパティーを選択します。
2. 「**プロパティーを選択**」フィールドで、ドロップダウン・メニューを使用して、条件値を持たせるプロパティーを選択します。
3. 「**条件演算子**」フィールドで以下のいずれかを選択します。

   == (等しい)

   != (等しくない)

   > (より大)

   < (より小)

   >= (より大か等しい)

   <= (より小か等しい)
4. 「**条件値**」フィールドで、依存プロパティーをテンプレートに組み込むために必要な値を入力します。

5. 「**使用可能なプロパティー**」表示で依存プロパティーを強調表示させて矢印をクリックし、「**依存プロパティー**」表示に移動させます。
6. 「**完了**」をクリックします。入力した情報が、Connector Configurator Express によって、Connector Configurator Express がインストールされている ¥bin ディレクトリーの ¥data¥app の下に XML 文書として保管されます。

パス名の設定

パス名の設定の一般規則を次に示します。

- Windows でのファイル名の最大長は 255 文字です。
- Windows では、絶対パス名は [Drive:][Directory]¥filename の形式に従う必要があります。例えば、C:¥WebSphereAdapters¥bin¥Data¥Std¥StdConnProps.xml のようになります。
- キュー名には、先行スペースや埋め込みスペースを使用できません。

新しい構成ファイルを作成

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

また、ファイルに対する拡張検証用のオペレーティング・システムを選択します。ツールバーには、「**ターゲット・システム**」というドロップ・リストがあります。ここから、プロパティーの拡張検証用のターゲット・オペレーティング・システムを選択できます。選択可能なオプションは、「Windows」、「i5/OS」、または「その他」(Windows ではない場合)、および「なし (拡張検証なし)」(拡張検証をオフに切り替え) です。始動時のデフォルトは Windows です。

Connector Configurator Express を始動するには、以下のステップを実行します。

- 「System Manager」ウィンドウで、「**ツール**」メニューから「**Connector Configurator Express**」を選択します。Connector Configurator Express が開きます。
- スタンドアロン・モードで Connector Configurator Express を起動します。

構成ファイルの拡張検証のオペレーティング・システムを設定するには、以下のステップを実行します。

- メニュー・バーの「**ターゲット・システム:**」ドロップ・リストをプルダウンします。
- 実行中のオペレーティング・システムを選択します。

「**ファイル**」>「**新規**」>「**コネクター構成**」を選択します。「新規コネクター」ウィンドウで、新規コネクターの名前を入力します。

また、統合ブローカーも選択する必要があります。選択したブローカーによって、構成ファイルに記述されるプロパティーが決まります。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。

- 「**Integration Broker**」フィールドで、ICS 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。

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. メニュー・バーの「**ターゲット・システム:**」ドロップ・リストを使用して、構成ファイルの拡張検証用のオペレーティング・システムを設定します (前述の『新しい構成ファイルを作成』を参照してください)。
2. 「**ファイル**」>「**新規**」>「**コネクター構成**」をクリックします。
3. 以下のフィールドを含む「**新規コネクター**」ダイアログ・ボックス表示されます。
   - **名前**

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

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

     ICS 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**

     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。
4. 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。
5. ファイルを保管するには、「**ファイル**」>「**保管**」>「**ファイルに**」をクリックするか、「**ファイル**」>「**保管**」>「**プロジェクトに**」をクリックします。プロジェクトに保管するには、System Manager が実行中でなければなりません。
   ファイルとして保管する場合は、「**ファイル・コネクターを保管**」ダイアログ・ボックスが表示されます。`*.cfg` をファイル・タイプとして選択し、「ファイル名」フィールド内に名前が正しいスペル (大文字と小文字の区別を含む) で表示されていることを確認してから、ファイルを保管するディレクトリーにナビゲートし、「**保管**」をクリックします。Connector Configurator Express のメッセージ・パネルの状況表示に、構成ファイルが正常に作成されたことが示されます。

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

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

既存ファイルの使用

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 Express でそのファイルを開き、構成を修正してから、再度保管する必要があります。

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 Express で、「**ファイル**」>「**開く**」>「**ファイルから**」をクリックします。
2. 「**ファイル・コネクターを開く**」ダイアログ・ボックス内で、以下のいずれかのファイル・タイプを選択して、使用可能なファイルを調べます。
   - 構成 (*.cfg)
   - 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 Express を始動します。
3. 「**ファイル**」>「**開く**」>「**プロジェクトから**」をクリックします。

構成ファイルの完成

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

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 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コネクター・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。表に、「**プロパティー名**」、「**値**」、「**タイプ**」、「**サブタイプ**」 (「タイプ」がストリングである場合)、「**説明**」、および「**更新メソッド**」が表示されます。
3. ここでファイルを保管するか、または 97 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。
4. 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。

   構成ファイルを作成する前に、プロパティーの拡張検証用のターゲット・オペレーティング・システムを選択することができる「**ターゲット・システム**」ドロップ・リストを使用しました。

   Connector Configurator Express 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 Express 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。

「**ターゲット・システム**」ドロップ・リストから「Windows」または「その他」の値を選択して拡張検証機能を使用する場合、システムはタイプだけでなくプロパティー・サブタイプも検証し、検証が失敗した場合は警告メッセージを表示します。

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

Connector Configurator Express 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
- セキュリティー

重要: Connector Configurator Express 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。
- プロパティーごとに「**更新メソッド**」フィールドが表示されます。これは、変更された値をアクティブにするためにコンポーネントまたはエージェントの再始動が必要かどうかを示します。この設定を構成することはできません。

標準コネクター・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。

   注: プロパティーのタイプが String である場合は、「サブタイプ」列にサブタイプ値が含まれている場合があります。このサブタイプは、プロパティーの拡張検証に使用されます。
3. 標準のプロパティーの値をすべて入力後、以下のいずれかを実行することができます。
   - 変更内容を破棄し、元の値を保持したままで Connector Configurator Express を終了するには、「**ファイル**」>「**終了**」をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら「**いいえ**」をクリックします。
   - Connector Configurator Express 内の他のカテゴリーの値を入力するには、そのカテゴリーのタブを選択します。「**標準のプロパティー**」(またはその他のカテゴリー) で入力した値は、次のカテゴリーに移動しても保持されます。ウィンドウを閉じると、すべてのカテゴリーで入力した値を一括して保管するかまたは破棄するかを確認するプロンプトが出されます。
   - 修正した値を保管するには、「**ファイル**」>「**終了**」をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら「**はい**」をクリックします。「ファイル」メニューまたはツールバーから「**保管**」>「**ファイルに**」をクリックする方法もあります。

特定の標準プロパティーの詳細を参照するには、「標準のプロパティー」タブ付きシートでそのプロパティーの「説明」列の記入項目を左マウス・ボタンでクリックします。全般ヘルプをインストールしてある場合は、右側に矢印ボタンが表示されます。ボタンをクリックすると、ヘルプ・ウィンドウが開かれ、標準プロパティーの詳細が表示されます。

注: ホット・ボタンが表示されない場合、そのプロパティーの全般ヘルプはありません。

全般ヘルプ・ファイルは、インストールされている場合、
<*ProductDir*>¥bin¥Data¥Std¥Help¥<*RegionalSetting*>¥ にあります。

コネクター固有の構成プロパティーの設定

コネクター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には「**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。

   注: プロパティーのタイプが String である場合は、「サブタイプ」ドロップ・リストからサブタイプを選択できます。このサブタイプは、プロパティーの拡張検証に使用されます。
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. 特定のプロパティーの詳細を参照するには、そのプロパティーの「説明」列の記入項目を左マウス・ボタンでクリックします。全般ヘルプをインストールしてある場合は、ホット・ボタンが表示されます。ホット・ボタンをクリックすると、ヘルプ・ウィンドウが開かれ、標準プロパティーの詳細が表示されます。

   注: ホット・ボタンが表示されない場合、そのプロパティーの全般ヘルプはありません。
5. 95 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。

全般ヘルプ・ファイルがインストールされており、AdapterHelpName プロパティーがブランクである場合、Connector Configurator Express は、
<*ProductDir*>¥bin¥Data¥App¥Help¥<*RegionalSetting*>¥ にあるアダプター固有の全般ヘルプ・ファイルを指します。それ以外の場合、Connector Configurator Express は、
<*ProductDir*>¥bin¥Data¥App¥Help¥<*AdapterHelpName*>¥<*RegionalSetting*>¥ にあるアダプター固有の全般ヘルプ・ファイルを指します。標準プロパティーに関する付録で AdapterHelpName プロパティーの説明を参照してください。

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

「コネクター固有プロパティー」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

更新メソッドについては、標準プロパティーに関する付録で 61 ページの『標準コネクター・プロパティーの概要』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 Express の「**サポートされているビジネス・オブジェクト**」タブを使用します。汎用ビジネス・オブジェクトと、アプリケーション固有のビジネス・オブジェクトの両方を指定する必要があり、またそれらのビジネス・オブジェクト間のマップの関連を指定することが必要です。

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・リストが表示されます。
2. 追加するビジネス・オブジェクトをクリックします。
3. ビジネス・オブジェクトの「**エージェント・サポート**」(以下で説明) を設定します。
4. 「Connector Configurator Express」ウィンドウの「ファイル」メニューで、「**プロジェクトに保管**」をクリックします。追加したビジネス・オブジェクト定義に

指定されたサポートを含む、変更されたコネクター定義が、System Manager の ICL (Integration Component Library) プロジェクトに保管されます。

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

1. ビジネス・オブジェクト・フィールドを選択するため、そのビジネス・オブジェクトの左側の番号をクリックします。
2. 「Connector Configurator Express」ウィンドウの「**編集**」メニューから、「**行を削除**」をクリックします。リスト表示からビジネス・オブジェクトが除去されます。
3. 「**ファイル**」メニューから、「**プロジェクトに保管**」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 Express」ウィンドウでは、「エージェント・サポート」を選択しても問題ないかどうかの検証は行われません。

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

ほとんどのコネクターの場合、選択可能な項目は「最大限の努力」のみです。

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

関連付けられたマップ

各コネクターは、現在 WebSphere InterChange Server Express 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 Express」ウィンドウの「**ファイル**」メニューから「**プロジェクトに保管**」を選択して変更を保管した後に、このリストに反映されます。

- **関連付けられたマップ**

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

- **明示的バインディング**

  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 Express」ウィンドウの「**ファイル**」メニューで、「**プロジェクトに保管**」をクリックします。
4. プロジェクトを 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。

セキュリティー

Connector Configurator Express の「**セキュリティー**」 タブを使用して、メッセージに対してさまざまなプライバシー・レベルを設定できます。この機能を使用できるのは、DeliveryTransport プロパティーが JMS に設定されている場合のみです。

デフォルトでは、プライバシーはオフになっています。使用可能にするには、「**プライバシー**」ボックスにチェック・マークを付けます。

「**鍵ストア・ターゲット・システムの絶対パス名**」は、次のとおりです。

- Windows の場合:
  `<ProductDir>¥connectors¥security¥<connectorname>.jks`

このパスとファイルは、コネクターを開始するシステム、つまりターゲット・システム上にある必要があります。

右側にある「参照」ボタンを使用できるのは、ターゲット・システムが、現在稼働中のシステムである場合のみです。「**プライバシー**」が使用可能で、メニュー・バーの「**ターゲット・システム**」が Windows に設定されていない限り、これはグレー表示されます。

3 つのメッセージ・カテゴリー (全メッセージ、全管理メッセージ、および全ビジネス・オブジェクト・メッセージ) に対して、「**メッセージのプライバシー・レベル**」を次のように設定できます。

- “” はデフォルトです。これは、メッセージ・カテゴリーに対してプライバシー・レベルが設定されていない場合に使用します。
- none
  これは、デフォルトと同じではありません。これは、メッセージ・カテゴリーに対してプライバシー・レベルを故意になしに設定する場合に使用します。
- integrity
- privacy
- integrity_plus_privacy

鍵の保守機能を使用すると、サーバーおよびアダプターに対して公開鍵を生成、インポート、およびエクスポートできます。

- 「**鍵の生成**」を選択すると、「鍵の生成」ダイアログ・ボックスが表示され、鍵を生成する keytool のデフォルトが示されます。
- 鍵ストア値は、デフォルトでは、「セキュリティー」タブの「**鍵ストア・ターゲット・システムの絶対パス名**」に入力した値になります。
- 「OK」を選択すると、記入項目が検証され、鍵証明書が生成されて、出力が「Connector Configurator Express」ログ・ウィンドウに送信されます。

証明書をアダプター鍵ストアにインポートする前に、サーバーの鍵ストアからエクスポートする必要があります。「**アダプター公開鍵のエクスポート**」を選択すると、「アダプター公開鍵のエクスポート」ダイアログ・ボックスが表示されます。

- エクスポート証明書は、デフォルトでは、ファイル拡張子が <filename>.cer であることを除き、鍵ストアと同じ値になります。

「**サーバー公開鍵のインポート**」を選択すると、「サーバー公開鍵のインポート」ダイアログ・ボックスが表示されます。

- インポート証明書は、デフォルトでは、<*ProductDir*>¥bin¥ics.cer になります (ファイルがシステムに存在する場合)。
- インポート証明書関連はサーバー名でなければなりません。サーバーが登録されている場合は、ドロップ・リストからそれを選択できます。

「**アダプター・アクセス制御**」機能を使用できるのは、DeliveryTransport の値が IDL である場合のみです。デフォルトでは、アダプターはゲスト ID を使用してログインします。「**ゲスト ID の使用**」ボックスにチェック・マークが付けられていない場合は、「**アダプター ID**」フィールドと「**アダプター・パスワード**」フィールドを使用できます。

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 Express は、そのファイルに含まれるロギングとトレースに関する値をデフォルト値として使用します。これらの値は、Connector Configurator Express 内で変更できます。

ログとトレースの値を変更するには、以下の手順を実行します。

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

     注: STDOUT オプションは、Windows プラットフォームで実行しているコネクターの「**トレース/ログ・ファイル**」タブでのみ使用できます。
   - ファイルに:
     ログ・メッセージまたはトレース・メッセージを指定されたファイルに書き込みます。ファイルを指定するには、ディレクトリー・ボタン (省略符号) をクリックし、指定する格納場所に移動し、ファイル名を指定し、「**保管**」をクリックします。ログ・メッセージまたはトレース・メッセージは、指定した場所の指定したファイルに書き込まれます。

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

データ・ハンドラー

データ・ハンドラー・セクションの構成が使用可能となるのは、DeliveryTransport の値に JMS を、また ContainerManagedEvents の値に JMS を指定した場合のみです。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。

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 Express 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 Express のタイトル・バーには、InterChange Server Express が現在使用しているブローカー・モードが常に表示されます。

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

- System Manager から、統合コンポーネント・ライブラリーに *.con 拡張子付きファイルとして保管します。
- System Manager から、指定したディレクトリーに *.con 拡張子付きファイルとして保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに *.cfg 拡張子付きファイルとして保管します。デフォルトでは、このファイルは ¥WebSphereAdapters¥bin¥Data¥App に保管されます。

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、インプリメンテーション・ガイドを参照してください。

構成の完了

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

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

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

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

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

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

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

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

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

                    <DefaultValue>en_US</DefaultValue>
            </ValidValues>
    </Property>
```

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、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 Corporation
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。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。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。著作権使用許諾: 本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

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
CICS
CrossWorlds
DB2
DB2 Universal Database
i5/OS
IMS
Informix
iSeries
Lotus
Lotus Domino
Lotus Notes
MQIntegrator
MQSeries
MVS
OS/400
Passport Advantage
SupportPac
WebSphere
z/OS
Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。
MMX および Pentium は、Intel Corporation の米国およびその他の国における商標です。
Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。
Linux は、Linus Torvalds の米国およびその他の国における商標です。
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 Server Express and Express Plus には、Eclipse Project (http://www.eclipse.org/) により開発されたソフトウェアが含まれています。

WebSphere Business Integration Server Express バージョン 4.4、および WebSphere Business Integration Server Express Plus バージョン 4.4

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