IBM WebSphere Business Integration Adapters

Adapter for EJB ユーザーズ・ガイド

バージョン 1.0.x

IBM WebSphere Business Integration Adapters

Adapter for EJB ユーザーズ・ガイド

バージョン 1.0.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、 95 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 Adapter for EJB バージョン 1.0.x、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： IBM WebSphere Business Integration Adapters
Adapter for EJB User Guide
Version 1.0.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2刷 2004.3

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

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

目次

本書について

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

本書では、IBM WebSphere Business Integration Adapter for EJB 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 EJB 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

本製品とともに提供される一連の資料では、WebSphere Business Integration Adapter のどのインストールにも共通の機能とコンポーネントを説明するとともに、特定のコンポーネントに関する参考資料についても記載しています。

以下のサイトから、関連資料をインストールすることができます。

- 一般的なアダプター情報が必要な場合、アダプターを WebSphere Message Broker (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) とともに使用する場合、およびアダプターを WebSphere Application Server とともに使用する場合は、以下のサイトを参照してください。
  - http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- アダプターを InterChange Server とともに使用する場合は、以下のサイトを参照してください。
  - http://www.ibm.com/websphere/integration/wicserver/infocenter
  - http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- Message Broker (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) の詳細については、以下のサイトを参照してください。

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

- WebSphere Application Server の詳細については、以下を参照してください。
  – http://www.ibm.com/software/webservers/appserv/library.html

これらのサイトには、資料をダウンロード、インストール、表示するための簡単な手順が示されています。

表記上の規則

本書は下記の規則に従って編集されています。

`Courier フォント`	コマンド名、ファイル名、入力情報、システムが画面に出力した情報など、記述されたとおりの値を示します。
太字	初出語を示します。
斜体、*斜体*	変数名または相互参照を示します。
青のアウトライン	青のアウトラインは、マニュアルをオンラインで表示するときにのみ見られるもので、相互参照用のハイパーリンクを示します。 アウトラインの内側をクリックすることにより、参照先オブジェクトにジャンプできます。
{ }	構文の記述行の場合、中括弧 {} で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
[ ]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプションのパラメーターです。
...	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、`option[,...]` は複数のオプションをコンマで区切って入力できることを意味します。
< >	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・パスに円記号 (¥) を使用します。UNIX システムの場合には、円記号 (¥) はスラッシュ (/) に置き換えてください。製品のパス名はすべて、使用システムでコネクターがインストールされたディレクトリーを基準とする相対パス名です。
`%`*`text`*`%` および `$`*`text`*	% 記号で囲まれたテキストは、Windows ™ の `text` システム変数またはユーザー変数の値を示します。UNIX 環境での同等の表記は `$`*`text`* です。これは、UNIX 環境変数 *`text`* 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。

本リリースの新機能

バージョン 1.0.x の新機能

2004 年 2 月:

- ロケール依存データに関する情報は、 1 ページの『アダプター環境』に移動しました。
- 9 ページの『動詞の処理』のセクションでは、ブランクの動詞 ASI について、リモート Enterprise Bean の場合と Java オブジェクトの場合での違いに関する情報が更新されました。
- 11 ページの『コネクターのファイル構造』のセクションでは、インストール時にシステムにコピーされる追加のファイル名に関する情報が更新されました。製品をインストール済みの場合、再インストールはしないでください。前回インストーラーを実行したときに、ファイルは既にシステムにコピーされているからです。
- 24 ページの『始動ファイルの構成』のセクションでは、新規のサンプル・ファイル名に関する情報が更新されました。
- 30 ページの『コネクター・ビジネス・オブジェクトの構造』のセクションでは、メソッドとビジネス・オブジェクト属性に関する情報が更新されました。
- 34 ページの『属性レベル ASI』のセクションでは、アプリケーション固有の情報 (`AppSpecificInfo` プロパティー) など、ビジネス・オブジェクト属性やそのプロパティーに関する情報が更新されました。
- `MusicCart` Enterprise Bean サンプル・ファイルの実行に関する情報は、 38 ページの『アダプターの実行によるサンプル・ビジネス・オブジェクトの起動』のセクションに記載されています。
- 43 ページの『ODA の始動』のセクションでは、始動ファイルの `CLIENT JARPATH` 変数に関する情報が更新されました。

2003 年 12 月:

- バージョン 1.0.x は、Adapter for EJB の最初のリリースです。

第 1 章 概要

この章では、Connector for EJB の概要を説明します。この章の内容は次のとおりです。

Connector for EJB は、WebSphere Business Integration Adapter for EJB のランタイム・コンポーネントです。EJB 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。コネクターを使用すれば、WebSphere 統合ブローカーは、Enterprise JavaBeans (EJB) アーキテクチャーを使用して設計され、アプリケーション・サーバー上に配置された Enterprise Bean との間で、ビジネス・オブジェクトを交換できます。この処理を行うために、コネクターは、ブローカー内のビジネス・プロセスが 1 つ以上の Enterprise Bean ビジネス・メソッドにデータを渡して、戻り値を受信できるようにします。アダプターは単一方向であり、要求処理機能だけを備えています。イベント通知は実行しません。

コネクターは、コネクター・フレームワークとアプリケーション固有のコンポーネントの 2 つのコンポーネントからなります。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 EJB) 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

本書には、コネクター・フレームワークとアプリケーション固有のコンポーネントの両方に関する情報が記載されています。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。

アダプター環境

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

- 3 ページの『アダプターのプラットフォーム』
- 3 ページの『アダプターの依存関係』
- 3 ページの『ロケールに依存するデータ』

ブローカーの互換性

アダプターが使用するアダプター・フレームワークは、アダプターと通信する統合ブローカーのバージョンとの互換性を備えている必要があります。Adapter for EJB バージョン 1.0.x は、以下のバージョンのアダプター・フレームワークおよび統合ブローカーでサポートされています。

- アダプター・フレームワーク:
  - WebSphere Business Integration Adapter Framework バージョン 2.10、2.2.0、2.3.0、2.3.1、2.4.0
- 統合ブローカー:
  - WebSphere InterChange Server バージョン 4.1.1、4.2.0、4.2.1、4.2.2
  - WebSphere MQ Integrator バージョン 2.1
  - WebSphere MQ Integrator Broker バージョン 2.1.0
  - WebSphere Business Integration Message Broker バージョン 5.0
  - WebSphere Application Server Enterprise バージョン 5.0.2 (WebSphere Studio Application Developer Integration Edition、バージョン 5.0.1 使用)

例外については、「*リリース情報*」を参照してください。

注: 統合ブローカーのインストール方法およびその前提条件については、以下の資料を参照してください。

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

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

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

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

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

アダプターの規格

このアダプターは、EJB 2.1 仕様に記載されています。したがって、WebSphere Application Server、バージョン 5.0 など、この規格に基づく J2EE 準拠のアプリケーション・サーバーと互換性があります。

このアダプターは、Entity Bean および Session Bean をサポートしています。Message-driven Bean は、このリリースではサポートされていません。

アダプターは、Java Authentication and Authorization Service (JAAS)、リリース 1.0 に対応しています。

EJB ソフトウェア・アーキテクチャーについては、http://java.sun.com/products/ejb/ を参照してください。

アダプターのプラットフォーム

このアダプターは、次のプラットフォームで動作します。

- Windows 2000
- Solaris 7、8
- HP-UX 11i
- AIX(R) 5.1、5.2

アダプターの依存関係

EJB アダプター用 ODA には `j2ee.jar` ファイルが必要です。このファイルは、アダプターとともに納入されます。

ロケールに依存するデータ

コネクターは、2 バイト文字セットをサポートする EJB インターフェースへの 2 バイト文字セットのデリバリーをサポートし、指定された言語でメッセージ・テキストを送信できるように国際化されています。ある文字コードを使用する場所から別の文字コード・セットを使用する場所へデータを転送する場合、コネクターは、そのデータの意味が伝わるように文字変換を実行します。

Java 仮想マシン (JVM) 内での Java ランタイム環境は、Unicode 文字コード・セットでデータを表します。Unicode には、最もよく知られた文字コード・セット (単一バイトとマルチバイトの両方) の文字エンコードが含まれています。WebSphere Business Integration システム内のほとんどのコンポーネントは、Java で書かれています。そのため、統合コンポーネント間でデータを転送する際に、ほとんどの場合文字変換は必要ありません。

用語

本書では、次の用語を使用します。

- **ASI (アプリケーション固有情報)**。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とビジネス・オブジェクト・レベルの両方に存在します。『*動詞 ASI*』も参照してください。
- **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) およびデータに対するアクション (作成操作や更新操作など) 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、情報の交換やアクションの起動を実行するためにビジネス・オブジェクトを使用します。

- **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するメソッドや要求ビジネス・オブジェクトをアプリケーション操作に変換するメソッドを含むコネクター・コンポーネント。
- **配置記述子**。JAR ファイル内でどのクラスが Bean を作成するか、Bean をどのように配置するか、各 Bean を実行時にどのように管理するかについて構造的なアプリケーション・アセンブリー情報を提供する、JAR (Java Archive) ファイルにパッケージされた XML ファイル。Enterprise Bean をエンタープライズ・アプリケーション・サーバーに配置すると、配置記述子が読み取られ、編集用にプロパティーが表示されます。Bean を配置したユーザーは、必要に応じて設定を変更および追加できます。配置情報を確認したら、それを使用してサーバーへの Enterprise Bean の配置に必要なサポート・インフラストラクチャー全体を生成します。
- **EJB コンテナー**。ライフ・サイクル管理、セキュリティー、配置、およびランタイム・サービスを提供することにより、Bean とアプリケーション・サーバーとの対話を管理します。このプログラマチック・エンティティーはアプリケーション・サーバー上に存在するため、クライアント・サイドではなくサーバー・サイドでキャッシングを処理します。コンテナーは、クライアント・アプリケーションによる直接アクセスから Enterprise Bean を隔離します。クライアント・アプリケーションが Enterprise Bean 上でリモート・メソッドを呼び出すと、コンテナーはまずその呼び出しを遮断し、クライアントが Bean 上で実行するすべての操作に対してパーシスタンス、トランザクション、およびセキュリティーが正しく適用されていることを確認します。Bean が使用されていない場合、コンテナーは、別のクライアントが再利用できるようにそれをプール内に置くか、あるいはメモリーから除去します。除去する場合、クライアント上のリモート参照はそのまま残るので、必要なときに元に戻すことができます。クライアントがリモート参照でメソッドを呼び出すと、EJB コンテナーは要求にサービスを提供する Bean だけを復元するので、クライアント・アプリケーションはプロセス全体を認識しません。Enterprise Bean は、EJB コンテナー外では機能しません。
- **EJB ホーム・オブジェクト**。Enterprise Bean のライフ・サイクル操作 (作成、削除、検索) を提供するオブジェクト。EJB ホーム・オブジェクトは Enterprise Bean のホーム・インターフェースをインプリメントし、コンテナーの配置ツールによって EJB ホーム・オブジェクトのクラスが生成されます。クライアントは Java Naming and Directory Interface (JNDI) を使用して、EJB ホーム・オブジェクトを検出します。次に、クライアントは EJB ホーム・オブジェクトを参照して、EJB オブジェクトでライフ・サイクル操作を実行します。
- **Enterprise Bean**。ホーム・インターフェース、リモート・インターフェース、および *Bean* クラス からなる Enterprise JavaBeans コンポーネントです。ホーム・インターフェースはコンポーネントのライフ・サイクル・メソッド (作成、削除、検索) を表し、リモート・インターフェースは Bean のビジネス・メソッドを表します。Bean クラスはリモート・インターフェースで定義されたビジネス・メソッドをインプリメントするので、Bean のキー・エレメントとなります。

  Enterprise Bean は JAR ファイルにパッケージされ、エンタープライズ・アプリケーション・サーバーに配置され、EJB コンテナーによって管理されます。EJB JAR ファイルには、1 つ以上の Enterprise Bean と配置記述子が含まれています。Enterprise Bean は、EJB コンテナー外では機能しません。

- **Enterprise Bean クラス**。Enterprise Bean のビジネス・メソッドをインプリメントします。
- **Enterprise JavaBeans (EJB)**。EJB を使用すれば、オブジェクト指向、分散型、トランザクション対応の、安全でポータブルなエンタープライズ・レベル Java アプリケーションを迅速かつ簡単に開発できます。EJB 仕様では、プログラミング・モデルが必須です。つまり、規則またはプロトコルと、EJB API を構築するクラスおよびインターフェースのセットが必要です。EJB プログラミング・モデルは、Enterprise Bean およびサーバー開発者に開発用の共通プラットフォームを提供します。
- **Entity Bean**。名詞として表現できるビジネス概念をモデル化する Bean。`Customer`、`Item`、および `Vendor` は、実世界のオブジェクトをモデル化していることから、Entity Bean の名前と考えられます。これらのオブジェクトは、通常、データベース内の永続レコードです。『*Session Bean*』も参照してください。
- **外部キー**。子ビジネス・オブジェクトを一意的に識別する値を持つ単純属性。通常、この属性は、子の基本キー値を組み込むことによって、子ビジネス・オブジェクトをその親に示します。Connector for EJB は、外部キーを使用して、プール可能な接続オブジェクトを指定します。
- **ホーム・インターフェース**。Enterprise Bean のリモート・インターフェースを作成、削除、検出するメソッドなど、Bean のライフ・サイクル・メソッドを定義します。また、特定の Bean インスタンスに固有でないビジネス・メソッドとしてのホーム・ビジネス・メソッドも定義します。
- **Java Authentication and Authorization Service (JAAS)**。サービスがユーザー ID を基にした認証およびアクセス制御を実行できるようにするセキュリティー・フレームワーク。プラグ可能認証モジュール (PAM) 標準フレームワークの Java 版をインプリメントし、ユーザー・ベースの権限をサポートします。JAAS 権限により、どのコードが実行中であるかだけでなく、誰がそのコードを実行しているかにも基づいて許可を与えることができます。
- **Java Naming and Directory Interface (JNDI)**。より広範なサービスにアクセスするために一定の API を提供する標準の Java 拡張機能。Connector for EJB は、JNDI を使用して、エンタープライズ・アプリケーション・サーバーに配置された Enterprise Bean を検出します。サーバーは JNDI をサポートする必要があり、そのためには、サーバーは Enterprise Bean をディレクトリー構造に編成して、そのディレクトリー構造にアクセスするための JNDI ドライバー (サービス・プロバイダーと呼ぶ) を提供する必要があります。
- **ODA (Object Discovery Agent)**。アプリケーション内で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エレメントを「ディスカバー」することによって、ビジネス・オブジェクト定義を自動的に生成するツール。アダプターをインストールすると、ODA が自動的にインストールされます。Business Object Designer には、ODA にアクセスし、それを対話式に操作するためのグラフィカル・ユーザー・インターフェースが用意されています。
- **Per-call オブジェクト・プール**。単一の `doVerbFor` メソッド呼び出し中にあるメソッドから別のメソッドへ渡す必要のあるオブジェクトを保管するためのプログラマチック・エンティティー。保管オブジェクトは、オブジェクト (子または子以外) あるいは単純属性です。

- **リモート・インターフェース**。クライアントが呼び出す Bean のビジネス・メソッドを定義します。これは、Bean が作業を実行するために外部に示すメソッドです。リモート・インターフェースは、ビジネス上の問題に焦点を当てるので、パーシスタンス、セキュリティー、並行性、トランザクションなどのシステム・レベルの操作のメソッドは組み込まれていません。コネクターは、処理の間、リモート・オブジェクト定義またはスタブを介して Enterprise Bean にアクセスします。
- **Session Bean**。クライアント・アプリケーションの拡張機能としてクライアントによって作成される Enterprise Bean。したがって、Session Bean は、計算やデータベースへのアクセスなど、クライアントのタスクとプロセスの管理に責任を負い、通常は単一のクライアント/サーバー・セッションの期間中にだけ存在します。Session Bean はトランザクション対応ですが、システムが破損した場合にリカバリーすることはできません。Session Bean オブジェクトは、ステートレスである場合もありますし、メソッドおよびトランザクション間で会話の状態を維持する場合もあります。Session Bean が状態を維持する場合に、メモリーからそのオブジェクトを除去しなければならないときは、EJB コンテナーがこの状態を管理します。ただし、Session Bean オブジェクト自体は独自の永続データを管理する必要があります。『*Entity Bean*』も参照してください。
- **動詞 ASI (アプリケーション固有情報)**。動詞 ASI は、ある動詞に対して、その動詞がアクティブである場合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します。動詞 ASI には、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名が含まれています。

EJB アダプターでの要求の処理

このセクションでは、7 ページの図 1 に示されているように、Connector for EJB が要求を処理する方法について説明します。

ここで説明する要求処理シナリオは、次のことを前提としています。

- WebSphere Application Server など、J2EE 準拠のエンタープライズ・アプリケーション・サーバーに、Enterprise Bean が配置されています。
- アプリケーション・サーバーがインストールされ、動作しています。
- コネクターが初期化されており、`ProviderURL` および `InitialContextFactory` コネクター固有プロパティーをロードします。これらのプロパティーは、JNDI 初期コンテキストを取得する上で役立ちます。これは、EJB サーバーへの接続を開始し、Enterprise Bean のホーム・インターフェースを検出します。これらのプロパティーは、コネクターのライフ・サイクルを通じてキャッシュされ、再利用されます。コネクター固有のプロパティーの詳細については、16 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。コネクターがセキュリティーを管理する方法の詳細については、21 ページの『セキュリティーの構成』を参照してください。

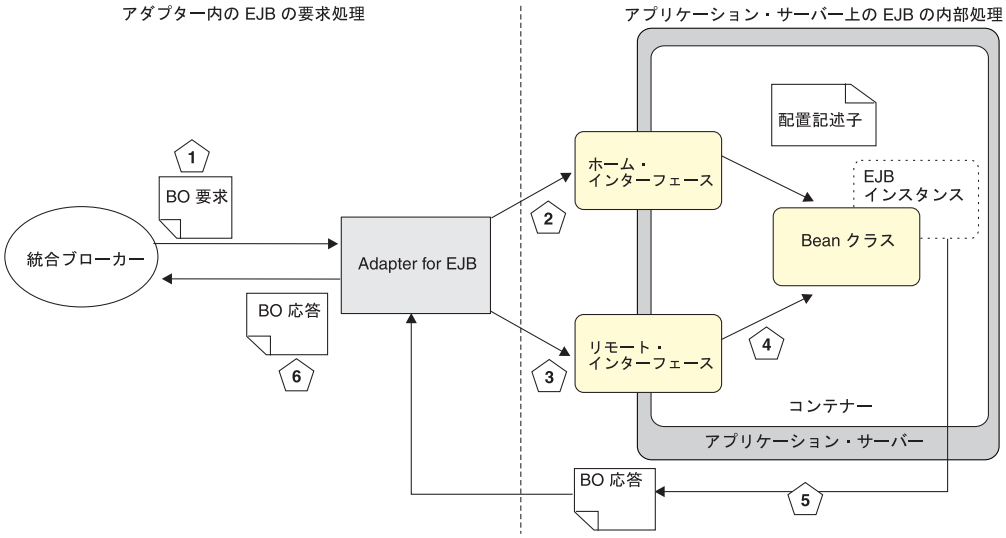


図 1. Connector for EJB の要求処理

Connector for EJB は、ビジネス・オブジェクト要求を以下の方法で処理します。

1. コネクターは、統合ブローカーからビジネス・オブジェクト要求を受信します。ビジネス・オブジェクトには、対応する Enterprise Bean に関する情報が含まれていて、例えば、その JNDI 名、ホーム・インターフェースおよびリモート・インターフェースのクラス名などがあります。
2. コネクターは、`InitialContextFactory` および `ProviderURL` プロパティーを使用して、アプリケーション・サーバーに配置された Enterprise Bean の検出およびアクセスの処理を開始します。`InitialContext` は、アプリケーション・サーバーでサポートされる、より大規模な JNDI API の一部です。これは、クライアント (コネクターなど) による任意の JNDI 検索の起点となります。`ProviderURL` はサービス・プロバイダーを示します。サービス・プロバイダーとは、Enterprise Bean のホーム・インターフェースの検出に使用される、アプリケーション・サーバー上の JNDI ドライバーです。ホーム・インターフェースは、Enterprise Bean のリモート・インターフェースを作成または検索するメソッドを提供します。
3. ホーム・インターフェースを検出したら、コネクターは、そのリモート・インターフェースを検索します。リモート・インターフェースには、クライアント (この場合はコネクター) が呼び出すことができる Enterprise Bean ビジネス・メソッドが定義されています。

   Entity Bean の場合、コネクターは creator タイプか finder タイプのどちらかのメソッドを使用しますが、これは、Enterprise Bean 開発者がどちらをインプリメントしたかに依存します (コネクターは、ビジネス・オブジェクト定義内のメタデータに基づいて、どちらを使用すべきかを認識します)。これらのメソッドはホーム・インターフェースに定義されており、Enterprise Bean インスタンスへのリモート/ローカル・インターフェース参照を取得するために、クライアントによって呼び出されます。
4. コネクターは、リモート・インターフェースを検出すれば、Enterprise Bean メソッドの呼び出しを開始できます。コネクターによって送信される親ビジネス・オ

ブジェクトには、リモート・インターフェースで定義された各メソッドの子ビジネス・オブジェクトが含まれています。子ビジネス・オブジェクトの属性は、対応する Enterprise Bean のリモート・メソッドのパラメーターにマップされます。Enterprise Bean は動的にロードされるので、そのメソッドはリフレクションによってディスカバーされ、呼び出されます。このステップの詳細は次のとおりです。

- コネクターの BO ハンドラーが、1 つ以上の属性名リストについて、親ビジネス・オブジェクトの動詞レベル ASI を調べます。リストは、セミコロンで区切られた属性名の番号付きリストです。
- 親ビジネス・オブジェクトの各属性には、リモート・インターフェースで起動されるメソッドを表す子ビジネス・オブジェクトが含まれています。つまり、動詞 ASI はメソッドのリストではなく属性のリストであり、各属性には値として、起動されるメソッドを表す子オブジェクトがあります。
- コネクターは、動詞 ASI にリストされたメソッドを、リストされている順番で実行します。

5. メソッドが実行され、値が EJB アプリケーション・サーバーから戻されると、コネクターはビジネス・オブジェクトに EJB オブジェクト・データをロードします。
6. コネクターは、EJB アプリケーション・サーバーから統合ブローカーへ、値を入力したビジネス・オブジェクトを戻します。

   また、コネクターは、元のオブジェクト要求が成功したか失敗したかを示すメッセージ (FAIL ステータス) を統合ブローカーに戻します。要求が成功した場合、コネクターは更新されたビジネス・オブジェクトをブローカーに戻します。

EJB トランザクション・サポート

コネクターは、EJB メソッドのトランザクション呼び出しをサポートしていません。ビジネス・プロセスでトランザクションが必要な場合は、コネクターに対しては 1 つの非トランザクション・メソッドを公開しながら、内部的にはアプリケーション・サーバーでトランザクションの一部としてターゲット EJB メソッドを呼び出すようなシン・ラッパー Bean を作成してください。

Adapter for EJB のセキュリティー

EJB セキュリティーはオプションの機能ですが、コネクターでこの機能を構成する場合、コネクターは認証およびアクセス制御のデータをアプリケーション・サーバーに提供する必要があります。そうすれば、コネクターはサーバー上に配置された任意のセキュア Bean にアクセスできます。Connector for EJB は、Java Authentication and Authorization Service (JAAS) を使用して EJB セキュリティーをインプリメントします。

JAAS は、サービスがユーザー ID を基にした認証およびアクセス制御を実行できるようにするセキュリティー・フレームワークです。プラグ可能認証モジュール (PAM) 標準フレームワークの Java 版をインプリメントし、ユーザー・ベースの権限をサポートします。JAAS 権限により、どのコードが実行中であるかだけでなく、誰がそのコードを実行しているかにも基づいて許可を与えることができます。

JAAS 認証は、プラグ可能な形式で実行されます。これにより、Java アプリケーションは基礎となる認証テクノロジーから独立した状態になるので、アプリケーション自体を変更することなく、新規認証テクノロジーや更新された認証テクノロジーをアプリケーションの下にプラグすることができます。

セキュリティーの詳細については、 21 ページの『セキュリティーの構成』を参照してください。

動詞の処理

コネクターは、各ビジネス・オブジェクトの動詞に基づいたブローカーによって、渡されるビジネス・オブジェクトを処理します。

コネクター・フレームワークは、ブローカーから要求を受信すると、要求ビジネス・オブジェクトのビジネス・オブジェクト定義に関連付けられた、business-object-handler 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、実行する動詞処理を決定することです。これは、操作要求を作成してアプリケーションへ送信するために、要求ビジネス・オブジェクトから情報を取得します。

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、このメソッドはビジネス・オブジェクト ASI を検索し、BO ハンドラーを呼び出します。BO ハンドラーは動詞 ASI を読み取り、それを一連の呼び出し可能関数に変換します。動詞 ASI は、その動詞に対して呼び出す必要のあるメソッドの番号付きリストです。オブジェクトの処理を成功させるためには、呼び出しを行う順序が重要です。

リモート Enterprise Bean の動詞 ASI がブランクである場合、BO ハンドラーは、引き数のないホーム・インターフェースで 1 つの creator メソッド (`Create()`) を検索して呼び出し、パラメーターにデータが取り込まれた最初のメソッドを呼び出します。コンストラクターの属性レベル ASI は `method_name=create` とします。

Java オブジェクトの動詞 ASI がブランクである場合、BO ハンドラーは、1 つのコンストラクターと、パラメーターにデータが取り込まれた 1 つのメソッドを検索します。コンストラクターの属性レベル ASI は `method_name=CONSTRUCTOR` とします。

リモート Enterprise Bean と Java オブジェクトのいずれの場合でも、データを取り込むことができるメソッドは 1 つだけです。動詞 ASI がブランクである場合に複数のメソッドにデータを取り込むと、コネクターはエラーを記録し、FAIL コードを戻します。エラー処理の詳細については、 55 ページの『エラー処理』を参照してください。動詞 ASI の詳細については、 32 ページの『動詞 ASI』を参照してください。

コネクターは特定の動詞をサポートしませんが、ユーザーは ODA を使用してカスタム動詞を構成できます。標準の既存の動詞は、Create、Retrieve、Update、および Delete です。これらには、Business Object Designer で実行している Object Discovery Agent (ODA) を使用して、任意のセマンティックの意味を与えることが

できます。ODA を使用して動詞にメソッド呼び出しシーケンスを割り当てる方法の詳細については、43 ページの『第 5 章 ビジネス・オブジェクトの作成と変更』を参照してください。

データ・ハンドラーの処理

データ・ハンドラーの使用はオプションです。データ・ハンドラーをコネクター・アーキテクチャーに組み込んで設計してある場合は、(EJB ASI の指定に従って) ビジネス・オブジェクト値を Enterprise Bean パラメーターに変換するために必要なあらゆるデータ・ハンドラー・クラスにコネクターがアクセスできる必要があります。Enterprise Bean メソッドは、XML、EDI、またはその他の WBI データ・ハンドラーがサポートする文書をリモート EJB メソッドへの引き数としてとることができます。

コネクターは、ビジネス・オブジェクトを受信すると、ビジネス・オブジェクト ASI を評価し、データ・ハンドラーを使用してビジネス・オブジェクトをデータに変換する必要があるかどうかを判別します。データ・ハンドラーがサポートするメッセージのビジネス・オブジェクト ASI には、値 `object_type=dataHandlerObject; mime_type=<text_value>` が含まれます。ここで `<text_value>` は、アダプターがデータの変換に使用するデータ・ハンドラーに対して定義された (データ・ハンドラー・メタオブジェクト内に指定された) 適切な MIME タイプを示します。

データ・ハンドラーがサポートする文書をパラメーターとして持つ method ビジネス・オブジェクトを検出した場合、コネクターは、ビジネス・オブジェクトを対応する文書に変換するデータ・ハンドラーを起動します。そして、データ・ハンドラーによって生成された文書を引き数としてメソッドに渡して、リモート Enterprise Bean メソッドを起動します。同様に、データ・ハンドラーを使用して処理する必要のある文書をメソッドが戻した場合、コネクターはデータ・ハンドラー値を使用して、メソッドから戻されたストリングをビジネス・オブジェクトに変換します。例えば、Enterprise Bean メソッドが XML または EDI 文書を戻した場合は、データ・ハンドラーを起動して、それを子ビジネス・オブジェクトに変換する必要があります。

データ・ハンドラーをサポートするには、`DataHandlerConfigMO` コネクター固有のプロパティーを構成する必要があります。このコネクター固有のプロパティーおよびその他のコネクター固有のプロパティーの詳細については、16 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。データ・ハンドラーの開発の詳細については、「データ・ハンドラー・ガイド」を参照してください。

第 2 章 アダプターのインストール

この章では、コネクターのインストール方法について説明します。この章の内容は次のとおりです。

- 『インストール作業の概要』
- 『コネクターのファイル構造』
- 13 ページの『インストール後の作業』

インストール作業の概要

Adapter for EJB をインストールするときは、必要なアダプター前提条件がご使用の環境に存在するかを確認し、統合ブローカーをインストールしてから、アダプターのインストールを実行する必要があります。

アダプターの前提条件の確認

アダプターをインストールする前に、アダプターをインストールおよび実行するための環境の前提条件がご使用のシステムですべて満たされていることを確認してください。詳細については、1 ページの『アダプター環境』を参照してください。

統合ブローカーのインストール

統合ブローカーのインストールでは、WebSphere Business Integration システムのインストールとブローカーの始動を実行します。この作業については、ご使用のブローカーの資料を参照してください。Connector for EJB がサポートしているブローカーの詳細については、2 ページの『ブローカーの互換性』を参照してください。

ブローカーのインストールの詳細については、ご使用のブローカーのインプリメンテーション資料を参照してください。

Adapter for EJB と関連ファイルのインストール

WebSphere Business Integration 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 Adapters Infocenter にある「*WebSphere Business Integration Adapters インストール・ガイド*」を参照してください。

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

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。また、コネクターを *ProductDir*¥connectors¥EJB ディレクトリーにインストールし、コネクターのショートカットを「スタート」メニューに追加します。*ProductDir* は、コネクターがインストールされるディレクトリーを表しています。

表 1 に、コネクターが使用するファイル構造と、インストーラーを使用してコネクターをインストールする場合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¥EJB¥BIA_EJB.jar	EJB コネクターに使用されるクラスだけを含みます。
¥connectors¥EJB¥start_EJB.bat	汎用コネクター (Windows 2000) の始動スクリプト。
¥connectors¥EJB¥start_EJB.sh	汎用コネクター (Unix) の始動スクリプト。
¥connectors¥messages¥BIA_EJBConnector.txt	コネクターのメッセージ・ファイル。
¥connectors¥EJB¥samples¥BIA_EJBConnector.cfg	サンプルの EJB コネクター構成ファイル。このファイルの使用の詳細については、 38 ページの『アダプターの実行によるサンプル・ビジネス・オブジェクトの起動』を参照してください。
¥connectors¥EJB¥samples¥BIA_PortConnector.cfg	サンプルのポート・コネクター構成ファイル。このファイルの使用の詳細については、 38 ページの『アダプターの実行によるサンプル・ビジネス・オブジェクトの起動』を参照してください。
¥connectors¥EJB¥samples¥SampleBOs¥	サンプルのビジネス・オブジェクト定義。これらのファイルの使用の詳細については、 38 ページの『アダプターの実行によるサンプル・ビジネス・オブジェクトの起動』を参照してください。
¥connectors¥EJB¥samples¥SampleMusicCartEJB¥src	サンプル Bean の EJB ソース・ファイル。サンプルの使用の詳細については、 38 ページの『アダプターの実行によるサンプル・ビジネス・オブジェクトの起動』を参照してください。
¥connectors¥EJB¥samples¥SampleMusicCartEJB¥BIA_MusicBeanSample.jar	サンプル・ファイルの実行時にアプリケーション・サーバーに配置した Enterprise Bean が格納されている EJB JAR ファイル。サンプルの使用の詳細については、 38 ページの『アダプターの実行によるサンプル・ビジネス・オブジェクトの起動』を参照してください。
¥repository¥EJB¥EJBConnectorTemplate	コネクター定義のためのテンプレート・ファイル。
¥connectors¥EJB¥dependencies¥j2ee.jar	EJB ODA が必要とする JAR ファイル。
¥ODA¥EJB¥BIA_EJBODA.jar	EJB ODA。
¥ODA¥EJB¥start_EJBODA.bat	ODA 始動ファイル (Windows 2000)。
¥ODA¥EJB¥start_EJBODA.sh	ODA 始動ファイル (Unix)。
¥ODA¥messages¥BIA_EJBODAAgent.txt	ODA のメッセージ・ファイル。
¥ODA¥messages¥BIA_EJBODAAgent_de_DE.txt	ODA のメッセージ・ファイル (ドイツ語テキスト・ストリング)。
¥ODA¥messages¥BIA_EJBODAAgent_en_US.txt	ODA のメッセージ・ファイル (米国英語テキスト・ストリング)。
¥ODA¥messages¥BIA_EJBODAAgent_es_ES.txt	ODA のメッセージ・ファイル (スペイン語テキスト・ストリング)。
¥ODA¥messages¥BIA_EJBODAAgent_fr_FR.txt	ODA のメッセージ・ファイル (フランス語テキスト・ストリング)。
¥ODA¥messages¥BIA_EJBODAAgent_it_IT.txt	ODA のメッセージ・ファイル (イタリア語テキスト・ストリング)。
¥ODA¥messages¥BIA_EJBODAAgent_ja_JP.txt	ODA のメッセージ・ファイル (日本語テキスト・ストリング)。
¥ODA¥messages¥BIA_EJBODAAgent_ko_KR.txt	ODA のメッセージ・ファイル (韓国語テキスト・ストリング)。
¥ODA¥messages¥BIA_EJBODAAgent_pt_BR.txt	ODA のメッセージ・ファイル (ポルトガル (ブラジル) 語テキスト・ストリング)。
¥ODA¥messages¥BIA_EJBODAAgent_zh_CN.txt	ODA のメッセージ・ファイル (中国語 (簡体字) 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_EJBODAAgent_zh_TW.txt	ODA のメッセージ・ファイル (中国語 (繁体字) テキスト・ストリング)。
¥repository¥EJB¥EJBConnectorTemplate	コネクターのリポジトリー定義。

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

アダプターをインストールしたら、実行する前に設定する必要があります。詳細については、 15 ページの『第 3 章 アダプターの構成』を参照してください。

第 3 章 アダプターの構成

この章では、アダプターの構成方法について説明します。この章の内容は次のとおりです。

構成作業の概要

インストールが完了したら、起動の前に、コネクターおよびビジネス・オブジェクトを構成する必要があります。

- **コネクターの構成**: コネクターの構成では、コネクターのセットアップと構成を行います。詳細については、『コネクターの構成』を参照してください。
- **ビジネス・オブジェクトの構成**: ビジネス・オブジェクトは、ODA (Object Discovery Agent) を介して構成します。ODA を使用すると、ビジネス・オブジェクト定義を生成できます。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・オブジェクトを調べ、ビジネス・オブジェクト属性に対応するオブジェクトのエレメントを「ディスカバー」し、情報を表すビジネス・オブジェクト定義を生成します。Business Object Designer には、Object Discovery Agent にアクセスし、それを対話式に操作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。

  ODA の使用の詳細については、43 ページの『第 5 章 ビジネス・オブジェクトの作成と変更』を参照してください。

コネクターの構成

コネクターの構成プロパティーには、標準構成プロパティーとコネクター固有の構成プロパティーという 2 つのタイプがあります。コネクターを実行する前に、Connector Configurator 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、77 ページの『付録 B. Connector Configurator』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。実行時セッション中に、1 つ以上のコネクター・プロパティーの値の変更が必要になることがあります。`AgentTraceLevel` など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、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。あるプロパ

ティーが動的 (即時に有効になる) か静的 (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。

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、すべてのコネク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、 59 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。

以下に示すプロパティーは、 59 ページの『付録 A. コネクターの標準構成プロパティー』にリストされていますが、Connector for EJB ではこれらのプロパティーを使用しません。

- `DuplicateEvent Elimination`
- `PollEndTime`
- `PollFrequency`
- `PollStartTime`

さらに、以下のプロパティーに関する注意事項があります。

- `Locale`: このコネクターは国際化されているので、この `Locale` プロパティーの値は変更できます。
- `ApplicationName`: コネクターを実行する前に、`ApplicationName` 構成プロパティーの値を指定する必要があります。

コネクター固有のプロパティー

コネクター固有の構成プロパティーは、コネクターが実行時に必要とする情報を提供します。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 を使用します。構成プロパティーを追加または変更するには、**「アプリケーション構成プロパティー」**タブをクリックします。詳細については、 77 ページの『付録 B. Connector Configurator』を参照してください。

表 2 に、コネクター固有の構成プロパティーと、その説明および可能な値を示します。これらのプロパティーはすべて非階層の String です。プロパティーの詳細については、 18 ページの図 2 のプロパティーのイメージなど、後続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
InitialContextFactory	初期コンテキスト・ファクトリーのクラス名。	`com.ibmwebsphere.naming.WsnInitialContextFactory`

表 2. コネクター固有のプロパティー (続き)

名前	指定可能な値	デフォルト値
ProviderURL	JNDI サービス・プロバイダーの URL。サービス・プロバイダーとは、Enterprise Bean が保管されているサーバー上のディレクトリーへのアクセスを可能にする、アプリケーション・サーバー上のドライバーです。	`corbaloc:iiop:localhost:2809`
DataHandlerConfigMO	データ・ハンドラー・メタオブジェクト。コネクター用にデータ・ハンドラーを定義した場合に、それをサポートするために使用されます。	`MO_DataHandler_Default`
LoginConfiguration	JAAS セキュリティー用の LoginModule クラスの名前。	デフォルト値は、ご使用のアプリケーション・サーバーによって決まります。WebSphere Application Server の場合、クラスは `WSLogin` です。 このプロパティーを使用できるのは、アプリケーション・サーバーで JAAS がサポートされている場合に、セキュア・アクセス Bean のインプリメントを選択したときだけです。
CallBackHandlerClass	JAAS セキュリティーに使用される、ユーザーがインプリメントした CallBackHandler インターフェース。	WebSphere Application Server の場合、クラスは `com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl` です。 このプロパティーを使用できるのは、アプリケーション・サーバーで JAAS がサポートされている場合に、セキュア・アクセス Bean のインプリメントを選択したときだけです。
JAASUserName	JAAS セキュリティー・ユーザー名。	なし このプロパティーを構成するのは、アプリケーション・サーバーで JAAS がサポートされている場合に、セキュア・アクセス Bean のインプリメントを選択したときだけです。
JAASPassword	JAAS セキュリティー・パスワード。	なし このプロパティーを構成するのは、アプリケーション・サーバーで JAAS がサポートされている場合に、セキュア・アクセス Bean のインプリメントを選択したときだけです。
JAASRealm	JAAS セキュリティー・レルム名。	なし このプロパティーを構成するのは、アプリケーション・サーバーで JAAS がサポートされている場合に、セキュア・アクセス Bean のインプリメントを選択したときだけです。

18 ページの図 2 に、コネクター固有のプロパティーの階層関係を示します。

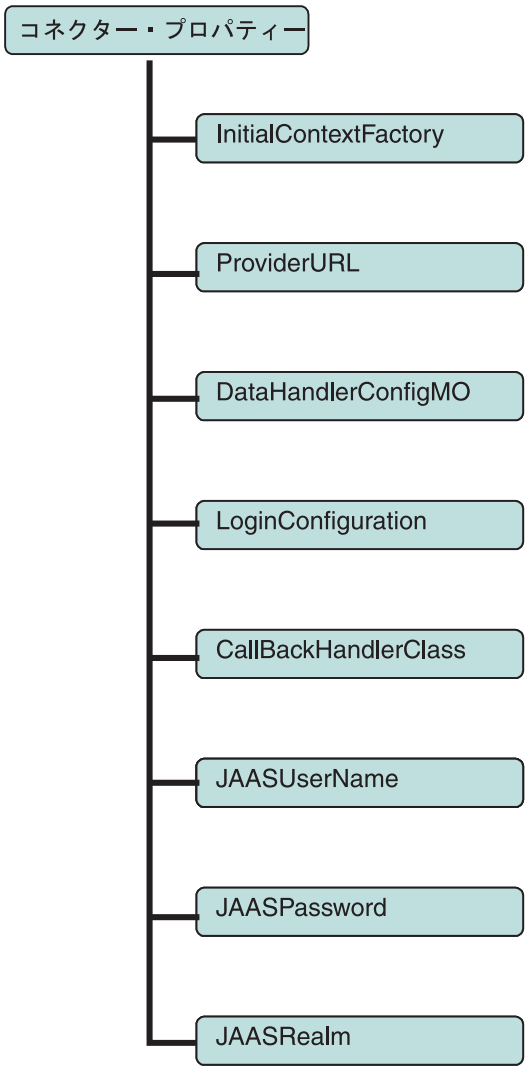


図 2. コネクター固有のプロパティーの階層

InitialContextFactory

初期コンテキスト・ファクトリーのクラス名。`InitialContext` は、エンタープライズ・アプリケーション・サーバーへの接続の開始および Enterprise Bean のホーム・インターフェースの検出に必要な JNDI インターフェースであり、Enterprise Bean の任意のクライアント検索の起点となります。コネクターは、`InitialContextFactory` プロパティーを `ProviderURL` プロパティーとともに使用して、JNDI 初期コンテキストを取得します。

ProviderURL

JNDI プロバイダーの URL。JNDI は、コネクターが Enterprise Bean に名前でアクセスできるようにします。コネクターはこの URL を使用して、EJB サーバー内で

動作している JNDI サーバーにリモート接続します。EJB サーバーへの接続後、コネクターは Enterprise Bean のホーム・インターフェースを見つけることができます。

DataHandlerConfigMO

トップレベル・データ・ハンドラー・メタオブジェクトの名前。データ・ハンドラーをコネクター・アーキテクチャーに組み込んで設計してある場合は、(EJB ASI の指定に従って) ビジネス・オブジェクト値を Enterprise Bean パラメーターに変換するために必要なあらゆるデータ・ハンドラー・クラスにコネクターがアクセスできる必要があります。Enterprise Bean メソッドは、XML、EDI、またはその他の WBI データ・ハンドラーがサポートする文書をリモート EJB メソッドへの引き数としてとることができます。データ・ハンドラーがサポートする文書をパラメーターとして持つ method ビジネス・オブジェクトを検出した場合、コネクターは、ビジネス・オブジェクトを対応する文書に変換するデータ・ハンドラーを起動します。そして、データ・ハンドラーによって生成された文書を引き数としてメソッドに渡して、リモート Enterprise Bean メソッドを起動します。

このプロパティーに値が入力されていない場合、コネクターは、適切なデータ・ハンドラーの起動に必要なメタオブジェクトを検出できません。コネクターでのデータ・ハンドラーの使用については、10 ページの『データ・ハンドラーの処理』を参照してください。

LoginConfiguration

認証テクノロジー・プロバイダーの `LoginModule` インターフェースをインプリメントするクラス。認証テクノロジー・プロバイダーは、`LoginModule` をインプリメントすることにより、認証可能なモジュールを介して特定の種類の認証を提供するので、アプリケーション自体を変更する必要はありません。通常は、アプリケーション・サーバーのログイン・クラスで `LoginConfiguration` および `LoginModule` インプリメンテーションが提供されます。JAAS を使用するセキュア・アクセス Bean をサポートしているアプリケーション・サーバーで、EJB セキュリティーに対してこのサービスをインプリメントする場合は、このプロパティーをアプリケーション・サーバーの LoginModule クラスの名前に設定します。

JAAS 準拠のアプリケーション・サーバーでは、コネクターが、JAAS 認証を処理する LoginModules を指定する LoginContext オブジェクトのインスタンスを生成することによって、認証プロセスを可能にします。クライアントの認証中に、LoginModule は、`JAASUserName` プロパティーおよび `JAASPassword` プロパティーで定義されたユーザー名とパスワードの入力を求め、それを確認します。

JAAS はコネクターの要件ではありませんが、`LoginConfiguration` プロパティーで値を指定すると、コネクターは Bean セキュリティーがインプリメントされていると見なすため、その場合は、`CallBackHandlerClass`、`JAASUserName`、`JAASPassword`、および `JAASRealm` プロパティーがすべて必要になります。

セキュリティーの構成の詳細については、21 ページの『セキュリティーの構成』および 8 ページの『Adapter for EJB のセキュリティー』を参照してください。

CallBackHandlerClass

アプリケーション・サーバーによって決まる JAAS インターフェース。これによりクライアントは、認証データをアプリケーション・サーバーに渡すことができます。このインターフェースが、基本となるセキュリティー・サービスに渡される CallBack ハンドラーをインプリメントすることにより、そのセキュリティー・サービスは、アプリケーションと対話し、クライアント (この場合はコネクター) からユーザー名やパスワードなどの特定認証データを検索することができます。 LoginModule は、CallBack ハンドラーを使用してクライアントと通信し、要求された認証データを取得します。コネクターのユーザー名とパスワードは、 JAASUserName および JAASPassword プロパティーに定義されています。

JAAS を使用するセキュア・アクセス Bean をサポートしているアプリケーション・サーバーで、EJB セキュリティーに対してこのサービスをインプリメントする場合は、この値を、ユーザーがインプリメントした CallBackHandler インターフェースに設定します。

JAAS はコネクターの要件ではありませんが、LoginConfiguration プロパティーで値を指定すると、コネクターは Bean セキュリティーがインプリメントされていると見なすため、その場合は、CallBackHandlerClass、JAASUserName、 JAASPassword、および JAASRealm プロパティーがすべて必要になります。

セキュリティーの構成の詳細については、 21 ページの『セキュリティーの構成』および 8 ページの『Adapter for EJB のセキュリティー』を参照してください。

JAASUserName

JAAS を使用するセキュア・アクセス Bean をサポートしているアプリケーション・サーバーで、EJB セキュリティーに対してこのサービスをインプリメントする場合は、この値をアプリケーション・サーバーで設定された JAAS セキュリティー・ユーザー名に設定します。

JAAS はコネクターの要件ではありませんが、JAAS を使用してセキュリティーを適切に構成するには、JAASUserName、LoginConfiguration、CallBackHandlerClass、 JAASPassword、および JAASRealm の各プロパティーで値を指定する必要があります。

セキュリティーの構成の詳細については、 21 ページの『セキュリティーの構成』および 8 ページの『Adapter for EJB のセキュリティー』を参照してください。

JAASPassword

JAAS を使用するセキュア・アクセス Bean をサポートしているアプリケーション・サーバーで、EJB セキュリティーに対してこのサービスをインプリメントする場合は、この値をアプリケーション・サーバーで設定された JAAS セキュリティー・パスワードに設定します。

JAAS はコネクターの要件ではありませんが、JAAS を使用してセキュリティーを適切に構成するには、JAASPassword、LoginConfiguration、CallBackHandlerClass、 JAASUserName、および JAASRealm の各プロパティーで値を指定する必要があります。

セキュリティーの構成の詳細については、『セキュリティーの構成』および 8 ページの『Adapter for EJB のセキュリティー』を参照してください。

JAASRealm

JAAS を使用するセキュア・アクセス Bean をサポートしているアプリケーション・サーバーで、EJB セキュリティーに対してこのサービスをインプリメントする場合は、この値を JAAS セキュリティー・レルム名に設定します。レルムは、特権または許可のセットへの 1 つ以上のユーザー・グループの JAAS マッピングです。

JAAS はコネクターの要件ではありませんが、JAAS を使用してセキュリティーを適切に構成するには、JAASRealm、LoginConfiguration、CallBackHandlerClass、JAASUserName、および JAASPassword の各プロパティーで値を指定する必要があります。

セキュリティーの構成の詳細については、『セキュリティーの構成』および 8 ページの『Adapter for EJB のセキュリティー』を参照してください。

セキュリティーの構成

コネクターの EJB セキュリティーはオプションであり、Enterprise Bean が配置されているアプリケーション・サーバーでセキュア Bean と選択したセキュリティー・プロトコルがサポートされている場合だけ使用できます。セキュリティーを使用可能にする場合は、配置された Bean にアクセスするため、ユーザー (この場合はコネクター) がアプリケーション・サーバーに認証データを提供する必要があります。Java Authentication and Authorization Service (JAAS) を使用して EJB コネクターでセキュリティーを使用可能にできるのは、アプリケーション・サーバーがこのセキュリティー・テクノロジーをサポートしている場合だけです。

セキュリティーの役割へのメソッド許可の割り当て

コネクターは、アクセス制御を使用して、各ユーザーが許可されたリソースにのみアクセスするようにします。アクセス制御は、特定のユーザーがシステム内で実行できることと実行できないことを規定したセキュリティー・ポリシーを適用します。EJB JAR ファイルでの Enterprise Bean のセキュリティーの役割によって、各セキュリティーの役割が起動できるホーム・インターフェースおよびリモート・インターフェースのメソッドが指定されます。配置記述子ファイルには、実行時にどの論理役割がどの Bean メソッドにアクセスできるかを宣言したタグが含まれています。

次のサンプル・コードは、配置記述子でセキュリティーの役割がメソッド許可にどのように割り当てられるかが示されます。配置記述子にリストされた各役割について、呼び出し可能なメソッドが Bean 名ごとにグループ化されています。

```
<method-permission>
        <role-name>payroll_department</role-name>
        <method>
                <ejb-name>EmployeeService</ejb-name>
                <method-name>*</method-name>
        </method>
</method-permission>
<method-permission>
        <role-name>employee</role-name>
```

```
        <method>
                <ejb-name>AardvarkPayroll</ejb-name>
                <method-name>findByPrimaryKey</method-name>
        </method>
        <method>
                <ejb-name>AardvarkPayroll</ejb-name>
                <method-name>getEmployeeInfo</method-name>
        </method>
        <method>
                <ejb-name>AardvarkPayroll</ejb-name>
                <method-name>updateEmployeeInfo</method-name>
        </method>
</method-permission>
<method-permission>
```

Java Authentication and Authorization Service (JAAS) の使用

アプリケーション・サーバーが Java Authentication and Authorization Service (JAAS) をサポートしている場合は、それによって、このサービスを使用する EJB セキュリティーをインプリメントできます。ユーザーまたはクライアント (この場合はコネクター) がアプリケーション・サーバーに配置されたセキュア Bean にアクセスしようとすると、JAAS はユーザーおよびそのユーザーの許可を確認します。その際、JAAS は、Enterprise Bean が配置されているアプリケーション・サーバーに別のレベルのセキュリティーを提供します。JAAS では、LoginContext、LoginModule、およびクライアントを認証するために使用される基本的なメソッドを記述したその他の J2EE クラスが必要です。これらのクラスの追加情報については、19 ページの『LoginConfiguration』を参照してください。

Connector for EJB はサーバーのクライアントとして動作するので、EJB 環境でセキュリティーが使用可能になっている場合、コネクターは、ユーザー認証データをユーザー ID (コネクターの JAASUserName 構成プロパティー) およびパスワード (JAASPassword 構成プロパティー) の形式で提供する必要があります。EJB コンテナーは、コネクターによって渡される ID または役割を調べ、それを配置記述子で定義されたメソッドに関連する ID オブジェクトのリストと比較します。コネクターからの呼び出し元 ID が、メソッドと関連する呼び出し元 ID と一致する場合は、メソッドを呼び出すことができます。

認証によって、ユーザー (この場合はコネクター) の ID が検証されます。ユーザーが認証システムを正常にパススルーすると、許可されたメソッドに自由にアクセスし、それを呼び出すことができます。

JAAS を使用する EJB セキュリティーを制御するコネクター・プロパティーは、LoginConfiguration、CallBackHandlerClass、JAASUserName、JAASPassword、および JAASRealm です。これらのプロパティーに入力する値は、ご使用のアプリケーション・サーバーによって異なります。これらのプロパティーがコネクターでどのように使用されるかについては、16 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。プロパティーに入力する値の詳細については、アプリケーション・サーバーの資料を参照してください。

コネクター始動ファイルでの JAAS の構成

JAAS を使用するコネクターでは、コネクター始動ファイル (start_EJB.bat または start_EJB.sh) に次の変更を加える必要があります。

- 始動ファイルに次のコマンドを入力して、JAAS 構成ファイルの名前と場所を指定します。このコマンドは、ファイルの最後の数行に表示されます。

  ```
  -Djava.security.auth.login.config=login.conf
  ```

  このサンプル・コマンドでは、JAAS 構成ファイルに login.conf という名前が付けられています。ただし、この名前は、アプリケーション・サーバー環境によって変わります。ご使用の環境での JAAS 構成ファイルの名前と場所の詳細については、ご使用のアプリケーション・サーバーの資料を参照してください。
- コネクターで JAAS API を使用するため、コネクター始動ファイルで SECURITY_SETTINGS セクションのコメントを外します。

  JAAS API は、EJB がサポートする任意のアプリケーション・サーバーによって提供される jaas.jar ファイルで入手できます。コネクターのクラスパスで、必ず jaas.jar を参照してください。

複数のコネクター・インスタンスの作成

コネクターの複数のインスタンスを作成する作業は、いろいろな意味で、カスタム・コネクターの作成と同じです。以下に示すステップを実行することによって、コネクターの複数のインスタンスを作成して実行するように、ご使用のシステムを設定することができます。次のようにする必要があります。

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

新規ディレクトリーの作成

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

```
ProductDir¥connectors¥connectorInstance
```

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

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

```
ProductDir¥repository¥connectorInstance
```

ビジネス・オブジェクト定義の作成

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

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

2. 初期コネクターのファイルは、次のディレクトリーに入っていなければなりません。

   ProductDir¥repository¥initialConnectorInstance

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

コネクター定義の作成

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

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

始動スクリプトの作成

始動スクリプトは以下のように作成します。

1. 初期コネクターの始動スクリプトをコピーし、次のようにコネクター・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、 23 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. 始動スクリプトのショートカットを作成します (Windows のみ)。
4. 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。

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

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

始動ファイルの構成

Connector for EJB を始動する前に、コネクター始動ファイルを構成する必要があります。

Windows プラットフォームのコネクターの構成を完了するには、start_EJB.bat ファイルを修正する必要があります。

1. start_EJB.bat ファイルを開きます。
2. 「SET JCLASSES...」で始まるセクションまでスクロールします。
3. ODA によって作成される JAR ファイルを指す JCLASSES 変数を編集します。例えば、ODA によって作成される JAR ファイルが c:¥WebSphereAdapters¥connectors¥EJB¥SampleBeans.jar である場合は、

JCLASSES 変数を
JCLASSES=.;%J_CLASSES%;c:¥WebSphereAdapters¥connectors¥EJB¥
SampleBeans.jar に設定します。

コネクターの始動

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

ProductDir¥connectors¥*connName*

ここで、*connName* はコネクターを示します。始動スクリプトの名前は、表 3 に示すように、オペレーティング・システム・プラットフォームによって異なります。

表 3. コネクターの始動スクリプト

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

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

- Windows システムで**「スタート」**メニューから。

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

    start_*connName connName brokerName* [-c*configFile*]
  - UNIX ベースのシステム:

    connector_manager_*connName* -start

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

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

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

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

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

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

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

コネクターの停止

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

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

    ```
    connector_manager_connName -stop
    ```

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

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

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

ログ・ファイルとトレース・ファイルの使用

コネクター・コンポーネントは、いくつかのレベルのメッセージのロギングと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、トレース・メッセージをログに記録します。エラー・メッセージと通知メッセージは、ログ・ファイルに記録され、トレース・メッセージとそれに対応するトレース・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・レベルとトレース・レベルの詳細については、55 ページの『第 6 章 トラブルシューティングとエラー処理』を参照してください。

Connector Configurator で、ログ・ファイル名とトレース・ファイル名、およびトレース・レベルを構成します。このツールの詳細については、77 ページの『付録 B. Connector Configurator』を参照してください。

ODA にはロギング機能がありません。エラー・メッセージはユーザー・インターフェースに直接送信されます。トレース・ファイルとトレース・レベルは、Business Object Designer で構成されます。このプロセスについては、45 ページの『エージェントの構成』を参照してください。ODA トレース・レベルは、57 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・オブジェクトについて

この章では、ビジネス・オブジェクトの構造、コネクターがビジネス・オブジェクトを処理する仕組み、およびビジネス・オブジェクトに関するコネクターの前提事項について説明します。

この章の内容は次のとおりです。

メタデータの定義

Connector for EJB 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、オブジェクトのデータ構造を記述したアプリケーション固有の情報として定義されます。メタデータは、コネクターが実行時にビジネス・オブジェクトの構築に使用する、ビジネス・オブジェクト定義の構成に使用されます。

コネクターをインストールしたら、それを実行する前に、ビジネス・オブジェクト定義を作成する必要があります。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従って処理を行います。これにより、コネクターは、コードを変更することなく、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。新規オブジェクトは、Business Object Designer の Object Discovery Agent (ODA) を使用して作成できます。既存のオブジェクトを変更するには、(ODA を経由せずに) Business Object Designer ディレクトリーを使用します。

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と、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送信されます。

コネクターには、サポートされるビジネス・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・オブジェクトの関係、データの形式についての想定があります。ビジネス・オブジェクトの構造は、対応する Enterprise Bean に対して定義された構造と一致していることが重要です。一致していないと、コネクターは、ビジネス・オブジェクトを正しく処理することができません。

ビジネス・オブジェクトの構造を変更する必要がある場合は、対応する Enterprise Bean に変更を加え、その変更を ODA に入力するため、JAR ファイルにエクスポートします。

ビジネス・オブジェクト定義の変更の詳細については、「*WebSphere Business Integration Adapters 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

コネクターは、Enterprise Bean が使用するビジネス・オブジェクトを処理します。このセクションでは、EJB コネクターによって処理されるビジネス・オブジェクトの構造に関する主要な概念について説明します。

メソッド

ODA は、Java クラス・ファイルに定義されているメソッドごとに、ビジネス・オブジェクトの属性を作成します (詳細については、『属性』および 34 ページの『属性レベル ASI』を参照してください)。

ODA がメソッドを基に作成する属性のタイプは、メソッド・パラメーターおよび戻りタイプを表す詳細な属性が格納されている子ビジネス・オブジェクトです。これらの属性は、EJB メソッドのパラメーターと同じ順序で表示されます。`Return_Value` 属性 (定義されるメソッドのタイプが `void` の場合には使用されない) は、引き数の順序の最後に置かれ、EJB メソッド呼び出しの結果を表します。子ビジネス・オブジェクト属性は、メソッド・パラメーターまたは戻り値のタイプに応じて、単純なタイプまたはオブジェクト・タイプ (複合) になります。このような属性のアプリケーション固有の情報 (ASI) には、公開されているリモート・メソッド名が含まれています。属性 ASI の詳細については、34 ページの『属性レベル ASI』を参照してください。

EJB コネクターには、ホーム・インターフェース creator メソッドのリストから取得した creator メソッドを持つビジネス・オブジェクトと、リモート・インターフェースに指定されている 1 つ以上のビジネス・オブジェクト・メソッドが必要です。ホーム・インターフェース上の各 `creator` メソッドと、リモート・インターフェースで定義された各メソッドに対して、ビジネス・オブジェクト内に属性が作成されます。

プロパティーまたはメソッド名に特殊文字が含まれている場合は常に、これらに対応する属性名が WebSphere Business Integration の形式に合うように修正されます。

属性

Java クラス・ファイルで定義された Enterprise Bean 内にある各 public メンバー変数、属性、およびメソッド・パラメーターごとに、対応するビジネス・オブジェクト属性が ODA によって生成されます。ODA は、インターフェースが格納されている JAR ファイルを使用して、リモートおよびローカルの Enterprise Bean にマップするビジネス・オブジェクト定義を作成します。

Bean クラス内の属性が単純属性ではなくオブジェクトである場合、ビジネス・オブジェクト (BO) 属性は、対応する Java クラスまたは EJB インターフェースと定義が一致する子オブジェクトにマップします。

ビジネス・オブジェクトは、フラットまたは階層です。フラット・ビジネス・オブジェクトには、単純属性、つまり、単一の値 (ストリングなど) を表す属性だけを格納できます。これは、子ビジネス・オブジェクトを指しません。階層ビジネス・オブジェクトには、単純属性と、属性値を含む子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列の両方が格納されます。

カーディナリティー 1 コンテナー・オブジェクト、または単一カーディナリティー関係が生じるのは、親ビジネス・オブジェクトの属性に単一の子ビジネス・オブジェクトが含まれている場合です。この場合、子ビジネス・オブジェクトは、レコードを 1 つだけ格納できるコレクションを表します。属性タイプは、子ビジネス・オブジェクトです。

カーディナリティー n コンテナー・オブジェクト、または複数カーディナリティー関係が生じるのは、親ビジネス・オブジェクトの属性に子ビジネス・オブジェクトの配列が含まれている場合です。この場合、子ビジネス・オブジェクトは、複数のレコードを格納できるコレクションを表します。属性タイプは、子ビジネス・オブジェクトの配列の属性タイプと同じです。

ビジネス・オブジェクト属性の追加情報については、 34 ページの『属性レベル ASI』を参照してください。

アプリケーション固有の情報

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固有の手順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は、定義内のアプリケーション固有情報が、コネクターが期待する構文と一致す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情報は、名前と値のペアとして表されます。これは、ビジネス・オブジェクト全体、各ビジネス・オブジェクト属性、および各動詞に対して指定できます。

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れに含まれる各オブジェクトの性質に関する基本情報を提供します。ビジネス・オブジェクトの ASI を表示するには、Business Object Designer でビジネス・オブジェクトを開き、「一般」タブをクリックします。ビジネス・オブジェクト・レベルの ASI は、**「ビジネス・オブジェクト・レベル・アプリケーション固有の情報」**フィールドに表示されます。この画面の詳細については、 43 ページの『第 5 章 ビジネス・オブジェクトの作成と変更』を参照してください。

32 ページの表 4 で、Enterprise Bean を表すビジネス・オブジェクトのビジネス・オブジェクト・レベル ASI について説明します。

注: メソッド、メソッド・パラメーター、およびメソッド戻り値を表すビジネス・オブジェクトの場合、ASI 名は認識されません。Enterprise Bean のメソッドに対して作成されるビジネス・オブジェクト属性の詳細については、 30 ページの『メソッド』を参照してください。

表 4. ビジネス・オブジェクト・レベル ASI

オブジェクト・レベル ASI	説明
object_type=RemoteEJB	オブジェクトが、アプリケーション・サーバーに配置されない標準 Java クラス (ローカル) ではなく、アプリケーション・サーバーに配置される Enterprise Bean (リモート) であることを示します。
jndi_name=<*JNDIName*>	Enterprise Bean の JNDI 名
home_name=<*className*>	ホーム・インターフェースのクラス名
proxy_class=<*className*>	リモート・インターフェースのクラス名

次の例では、WebSphere Application Server に配置された Enterprise Bean を使用してコネクターが処理する、ビジネス・オブジェクトのビジネス・オブジェクト・レベル ASI を示します。コネクターは、ブローカーからビジネス・オブジェクトを受信すると、まず構成された JNDI コンテキスト内の WsSamples/Account を検索して、アプリケーション・サーバー上のホーム・インターフェース (com.ibm.websphere.AccountBookHome) のインスタンスを見つけます。次に、コネクターはこのホーム・インスタンスを使用してリモート・インターフェース (com.ibm.websphere.AccountBook) の新規インスタンスを作成します。これを使用すると、EJB でメソッドを起動できます。

```
BO ASI=object_type = RemoteEJBObject
           proxy_class = com.ibm.websphere.AccountBook
           home_class = com.ibm.websphere.AccountBookHome
           jndi_name = WsSamples/Account
```

このコード例について説明します。

- BO ASI = object type は、オブジェクトがリモート・オブジェクトであることを表すために必要です。
- proxy_class = com.ibm.websphere.AccountBook は、リモート・スタブの名前を表します。
- home_class = com.ibm.websphere.AccountBookHome は、ホーム・インターフェースの名前です。
- jndi_name = WsSamples/Account は JNDI 名です。

標準 Java オブジェクト (ローカル) には、BO ASI = auto_load_or_write があります。

データ・ハンドラーがサポートするメッセージのビジネス・オブジェクト ASI には、値 object_type=dataHandlerObject; mime_type=<*text_value*> が含まれます。ここで <*text_value*> は、アダプターがデータの変換に使用するデータ・ハンドラーに対して定義された (データ・ハンドラー・メタオブジェクト内に指定された) 適切な MIME タイプを示します。

動詞 ASI

すべてのビジネス・オブジェクトに動詞が含まれています。動詞は、Enterprise Bean 上でどのメソッドを呼び出すかを示します。Adapter for EJB の場合、最初のメソッドは、対応する Bean のホーム・インターフェースにある creator メソッドとし、残りのメソッドは、その Bean のリモート・インターフェースにあるビジネス・プロセス・メソッドとします。

動詞 ASI には、属性名のシーケンスが含まれています。各属性名には、ビジネス・オブジェクト・ハンドラーが呼び出しを行うためのメソッドが含まれています。通常、呼び出されるメソッドは、(ビジネス・オブジェクトの親ではなく) オブジェクト自体に属しています。そのため、オブジェクトの動詞 ASI でメソッドを指定します。

呼び出されるメソッドがビジネス・オブジェクト階層内の親に属している場合は、メソッド名のプレフィックスとして PARENT タグを付けることにより、子からその親を参照できます。

例えば、図 3 は、ContactDetails が Contact の子オブジェクトで、Contact が PSRCustomerAccount の子オブジェクトであるビジネス・オブジェクト階層を示しています。

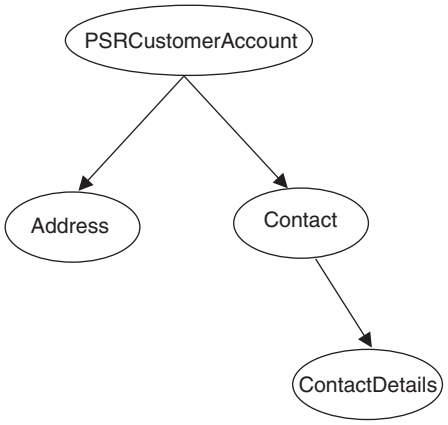


図 3. ビジネス・オブジェクト階層と動詞 ASI

PSRCustomerAccount に属すメソッドが ContactDetails ビジネス・オブジェクトで呼び出される場合、ContactDetails の動詞 ASI は次のようなビジネス・オブジェクト階層を示します。

```
PARENT.PARENT.<methodName>
```

一方、メソッドが Contact ビジネス・オブジェクトに属している場合は、ContactDetails の動詞 ASI を次のように設定する必要があります。

```
PARENT.<methodName>
```

階層内の親オブジェクトに属すメソッドだけを呼び出すことができます。親ビジネス・オブジェクトが子のメソッドを呼び出すことはできません。

コネクター開発者は、動詞に割り当てる EJB 操作を決めます。サポートされる動詞は次のとおりです。

- Create
- Delete
- Retrieve
- Update

所定のオブジェクトに対して、サポートされている 4 つの動詞 (Create、Retrieve、Delete、および Update) を指定し、各動詞のアクションとして n 個のメソッドを割り当てることができます。ここで n は、対応する Enterprise Bean 内のメソッドの数と同じです。

ビジネス・オブジェクトの動詞 ASI を表示するには、Business Object Designer でビジネス・オブジェクトを開き、「一般」タブをクリックします。動詞 ASI は、**「サポートされている動詞」**表にあります。この表には、サポートされている動詞とそれに対応する ASI が動詞ごとに記載されています。Business Object Designer と、ODA ウィザードのタブおよび画面の詳細については、43 ページの『第 5 章 ビジネス・オブジェクトの作成と変更』を参照してください。

属性レベル ASI

30 ページの『属性』に説明されているように、すべてのビジネス・オブジェクトには、これに対応する Enterprise Bean メソッドおよびプロパティーをマップする 1 組の属性があります。均一な構造を持つビジネス・オブジェクトは、単純属性、つまり単一値 (ストリングなど) を表す属性を格納し、子ビジネス・オブジェクトは指しません。階層ビジネス・オブジェクトには、単純属性と、属性値を含む子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列の両方が格納されます。

ビジネス・オブジェクト属性 ASI には、単純属性用と、子オブジェクトを含む複合属性用とがあります。複合属性の場合は、含まれる子がプロパティーかオブジェクトのメソッドかによって、ASI が異なります。ODA 生成のビジネス・オブジェクトへの EJB 構成体のマッピングについては、36 ページの表 8 を参照してください。

ビジネス・オブジェクト属性の ASI を表示するには、Business Object Designer でビジネス・オブジェクトを開き、「属性」タブをクリックします。ビジネス・オブジェクトの各属性は、**「名前」**列に表示されます。表 5、35 ページの表 6、および 36 ページの表 7 で説明する属性プロパティーは、残りの列に表示されます。これらのプロパティーに組み込まれているのは ASI で、ASI は**「アプリケーション固有の情報」**列に表示されます。この画面の詳細については、43 ページの『第 5 章 ビジネス・オブジェクトの作成と変更』を参照してください。

表 5 に、単純属性のビジネス・オブジェクト・プロパティーを示します。これらのプロパティーには、属性の ASI (AppSpecificInfo プロパティー) が組み込まれています。単純属性は、常に子以外です。例えば、ブール値、ストリング、整数値などです。

表 5. BO 属性プロパティー: 単純属性の場合

属性プロパティー	説明
Name	ビジネス・オブジェクト属性名を指定します。
Type	ビジネス・オブジェクト属性タイプを指定します。ビジネス・オブジェクト属性タイプへの EJB 構成体のマッピングの詳細については、36 ページの表 8 を参照してください。
MaxLength	使用されません。
IsKey	各ビジネス・オブジェクトには、少なくとも 1 つのキー属性が必要です。これは、属性のキー・プロパティーを true に設定することによって指定します。この属性は、コネクターではなく Business Object Designer によって使用されます。
IsForeignKey	オブジェクトを呼び出しごとのオブジェクト・プールに保管する必要があるかどうかをコネクターで確認することを指定します。

表 5. BO 属性プロパティー: 単純属性の場合 (続き)

属性プロパティー	説明
IsRequired	使用されません。
AppSpecInfo	元の Java タイプを保持します。このプロパティーの形式は次のとおりです。 property=<*propertyName*>, type=<*typeName*> property は、EJB 属性またはオブジェクト・メンバー変数の名前です。元の EJB オブジェクト・メンバー変数名を取り込むには、この名前と値のペアを使用します。 type は、単純属性の Java タイプです (EJB オブジェクト・メンバー変数)。例えば、type=java.lang.String のようになります。ビジネス・オブジェクト属性タイプへの EJB 構成体のマッピングの詳細については、36 ページの表 8 を参照してください。
DefaultValue	使用されません。

表 6 で、メソッドではない子オブジェクトを含む複合属性のエレメントについて説明します。これらのエレメントには、属性の ASI などがあります。

表 6. BO 属性プロパティー: メソッド以外の子オブジェクト属性の場合

属性プロパティー	説明
Name	ビジネス・オブジェクト属性名を指定します。
type	含まれているオブジェクトのタイプ。タイプがビジネス・オブジェクトである場合は、proxy に設定します。
ContainedObjectVersion	使用されません。
Relationship	子がコンテナー属性であることを指定します。Containment に設定します。
IsKey	使用されません。
IsForeignKey	使用されません。
Is Required	使用されません。
AppSpecificInfo	元の EJB アプリケーション・フィールド名を保持します。このプロパティーの形式は次のとおりです。 property=*propertyName*, use_attribute_value=<*(optional)BOName.AttributeName*>, type=<typeName> property は、EJB オブジェクト・メンバー変数の名前です。元の EJB オブジェクト属性名を取得するには、この名前と値のペアを使用します。メソッドへの引き数を持つ属性の場合は、property に値を設定しないでください。これは、この引き数に名前がなく、標準タイプの引き数にすぎないためです。 use_attribute_value は、ビジネス・オブジェクト名であり、形式は *BOName.AttributeName* です。この ASI を設定すると、コネクターが呼び出しごとのオブジェクト・プールからの属性にアクセスします。この値はビジネス・オブジェクトの作成時に ODA で設定するのではなく、Business Object Designer で設定します。type は、プロパティーの Java タイプです。ビジネス・オブジェクトへの EJB 構成体のマッピングについては、36 ページの表 8 を参照してください。
Cardinality	タイプが配列またはベクトルを表す場合は、*n* に設定します。それ以外の場合は、1 に設定します。

表 7 で、メソッドである子オブジェクトを含む複合属性のプロパティーについて説明します。これらのプロパティーには、属性の ASI などがあります。

表 7. BO 属性プロパティー: メソッドの子オブジェクト属性の場合

属性プロパティー	説明
Name	ビジネス・オブジェクト属性名。
type	ビジネス・オブジェクト。
Relationship	Containment に設定します。これは、これが子オブジェクトであることを示します。
IsKey	属性名が UniqueName に等しい場合は true に設定します。それ以外の場合は、false に設定します。
IsForeignKey	false に設定します。
Is Required	false に設定します。
AppSpecificInfo	公開されたリモート・メソッド・インターフェース名を保持します。この名前は、Enterprise Bean によって EJB サーバーに置かれるメソッド呼び出しの名前です。この属性の形式は次のとおりです。 method_name=<*remoteClassName.RemoteMethodName*> メソッドがコンストラクターである場合は、method_name=CONSTRUCTOR になります。
Cardinality	1 に設定します。

メソッドには引き数と戻り値があります。引き数と戻り値は、複合 (子オブジェクトを含む) または単純です。

マッピング属性: Enterprise JavaBeans (EJB) およびビジネス・オブジェクト

ここでは、JAR ファイルで定義された EJB 構成体と、それに対応するビジネス・オブジェクト属性のリストを示します。子ビジネス・オブジェクト以外のすべてのビジネス・オブジェクト属性のデータ型が String です。ビジネス・オブジェクトでは、Enterprise Bean のリモート・インターフェースに対してメソッドを起動したときに、属性の実際のデータ型を保持する ASI が使用されます。

ビジネス・オブジェクト ASI の詳細については、31 ページの『アプリケーション固有の情報』を参照してください。

表 8. オブジェクトのマッピング: Enterprise Bean をビジネス・オブジェクトへ

EJB 構成体	ビジネス・オブジェクト	属性 ASI タイプ =
JAR ファイル内で参照が検出されたすべてのクラス	Object	proxy_class=<*remote interface name*>
ブール値	Boolean	type=boolean/Boolean
char/Character	String	type=char/Character
Byte/Byte	String	type=byte/Byte
java.lang.String	String	type=string
Short/Short	Integer	type=short/Short

表 8. オブジェクトのマッピング: *Enterprise Bean* をビジネス・オブジェクトへ *(続き)*

EJB 構成体	ビジネス・オブジェクト	属性 ASI タイプ =
int/Integer	Integer	type=int/Integer
Long/Long	Integer	type=long/Long
float/Float	Float	type=float/Float
Double/double	Double	type=double/Double
java.math.BigDecimal	String	type=BigDecimal
クラス	Object	proxy_class=<*fully qualified class name*>
配列	Object 複数のカーディナリティーを持つ子ビジネス・オブジェクト	type=ArrayOf_<*datatype*> 例えば、type=ArrayOf_int
メソッド	Object Child BO	method_name=<*methodName*>
メソッド (引き数なしで戻り型が void)	String	method_name=<*methodName*>

注: 属性を参照解除しない場合は、ASI type=PlaceholderOnly を使用する必要があります。これは、この属性にデータを取り込まないようコネクターに指示します。ただし、外部キーとしてマークされている場合 (IsForeignKey が trueに設定されている場合) や、互換性のある属性を指す ASI use_attribute_value がある場合は、属性を複数呼び出しフローの一部として使用できます。

配列型

array 型については、次のことに注意してください。

- array 型を使用するには、type=ArrayOf_<*value*> の ASI を指定する必要があります。ここで、*value* は、 36 ページの表 8 にリストされている属性 ASI 値の 1 つです。例えば、type=ArrayOf_int は int 変数の配列を指定します。これらは、エレメントを含むカーディナリティー n ビジネス・オブジェクトにマップされます。
- Java の Object 配列 (Object[]) には、対応する ASI タイプ ArrayOf_proxy があります。配列のすべてのエレメントに対してプロキシー・オブジェクトの処理が実行されます。プロキシー配列が関数への引き数である場合は、メソッドの実行*前に* 配列内の全オブジェクトで動詞の処理が行われます。配列が戻り値である場合は、メソッドの実行*後に* 配列内の全オブジェクトで動詞の処理が行われます。
- サイズ変更された配列は、入力として使用できますが、出力としては使用できません。
- SafeArray は、入力としても戻り値としてもサポートされます。

アダプターの実行によるサンプル・ビジネス・オブジェクトの起動

これ以降では、製品に付属のサンプル・ファイルの実行について説明し、コネクターが JAR ファイルのサービスを起動してビジネス・オブジェクトを作成する方法を示す例を取り上げます。

- 『サンプル・ファイルの実行』
- 39 ページの『EJB JAR ファイル・コード』
- 40 ページの『ビジネス・オブジェクトのサンプル』

サンプル・ファイルの実行

サンプル・ファイルの格納先は、*ProductDir*¥connectors¥EJB¥samples ディレクトリーです。*ProductDir* は、コネクターのインストール先ディレクトリーを表しています。

サンプル・ファイルでは、ホーム・インターフェース、リモート・インターフェース、Bean、2 つのヘルパー・クラス RecordingHelper および CustomerHelper が用意されている簡単な Session Bean、MusicCart について説明しています。このサンプル・ファイルには、次の2 つのコネクター構成ファイルも収録されています。1つは BIA_EJBConnector.cfg で、ここにはサンプルを正常に実行するためのコネクター・プロパティー設定が記録されています。もう一方は BIA_PortConnector.cfg で、Test Connector のコネクター・プロパティーが格納されています。

MusicCart サンプル Bean には、Java オブジェクトを入力パラメーターとして使用し、オブジェクトを出力として返すメソッドがあります。この Bean には、オブジェクト配列を入力として使用し、アダプターによる配列の管理方法を示すメソッドもあります。このサンプルでは、アダプターによるデータ・ハンドラー処理の実行方法についても説明します。

このサンプル・ファイルに収録されているのは、ビジネス・オブジェクト・ファイル (BIA_SampleMusicCartBO.bo) と EJB ソース・ファイル (格納先は *ProductDir*¥connectors¥EJB¥samples¥SampleMusicCartEJB¥src) です。EJB ソース・ファイルは、EJB リモート・クラスおよびホーム・クラスが EJB の ODA によってどのようにビジネス・オブジェクトにマップされるかを示します。Test Connector からアプリケーション・サーバーに BIA_SampleMusicCartBO.bo を送信することにより、addRecording、getFirstRecordInfo、modifyMusicRequestUsingDataHandler、getAllRecordInfo の各メソッドの実行方法やコネクターによる要求処理の方法がわかります。

以下の手順では、WebSphere Application Server 5.0 に配置された Enterprise Bean とビジネス・オブジェクトとを交換するためにコネクターを実行していることを前提としています。

1. 11 ページの『第 2 章 アダプターのインストール』に説明されている方法で、Adapter for EJB をインストールします。
2. EJB JAR ファイル *ProductDir*¥connectors¥EJB¥samples¥SampleMusicCartEJB¥BIA_MusicBeanSample.jar を、WebSphere Application Server 5.0 のインスタンスに配置します。

3. *ProductDir*¥connectors¥EJB¥samples¥ ディレクトリーにある 2 つのコネクター構成ファイル (BIA_EJBConnector.cfg および BIA_PortConnector.cfg) を、InterChange Server などの統合ブローカーのリポジトリーにロードします。
4. *ProductDir*¥connectors¥EJB¥samples¥SampleBOs ディレクトリーから、サンプル・ビジネス・オブジェクトを統合ブローカー・リポジトリーにロードします。
5. アプリケーション・サーバーが実行されているマシンとは異なるマシンにリモート・クライアントをインストールする場合は、WebSphere Application Thin Client をインストールします。
6. コネクター始動ファイルを変更して、ステップ 2 (38 ページ) で配置した JAR ファイル (BIA_MusicBeanSample.jar) の場所を指すようにします。始動ファイル編集の詳細については、 24 ページの『始動ファイルの構成』を参照してください。
7. 始動ファイルで、アプリケーション・サーバー設定のコマンドのコメントを外します。このサンプルでは、WebSphere Application Server のコマンドのコメントを外す必要があります。このサンプルでは、WebSphere Application Server 5.0 に配置された Enterprise Bean とビジネス・オブジェクトとを交換するためにコネクターを実行していることが前提だからです。
8. WebSphere Application Server のインスタンスを始動します。
9. 25 ページの『コネクターの始動』に説明されている方法で、コネクターの実行を開始します。
10. Test Connector からアプリケーション・サーバーにファイル BIA_SampleMusicCartBO.bo を送信します。

コネクターが動作するにつれて、サンプル・ファイルで定義された Enterprise Bean メソッドがアダプターから実行され、リモートの WebSphere Application Server によって提供されたサービスが起動される様子を観察できます。

EJB JAR ファイル・コード

次のサンプル・コードは、MusicCartBean という EJB クラスのメソッドを定義する EJB JAR ファイルからの抜粋です。コード・サンプルの最後に定義されているメソッド getCustomer に注意してください。

注: このクラスのすべてのメソッドが、ここに示すサンプル・コードで定義されているわけではありません。このサンプルは、 38 ページの『サンプル・ファイルの実行』で説明されている大規模なファイルの一部です。

このコードに対応するビジネス・オブジェクトについては、 41 ページの図 4 に示します。

```
public class MusicCartBean implements SessionBean {
  CustomerHelper customerHelper;
  ArrayList shoppingList;
  RecordingHelper[] recordHelperArr;


   // EJB Methods
   public void ejbRemove() {}
   public void ejbActivate() {}
   public void ejbPassivate() {}
   public void setSessionContext (SessionContext ctx) {}
```

```
public void ejbCreate(String person,String password, String email)
    throws CreateException {
         if (person == null || person.equals("")) {
           throw new CreateException(
             "Name cannot be null or empty.");
         }
         else {
           customerHelper = new
             CustomerHelper(person, password, email);
           customerHelper.setName(person);
           customerHelper.setEmail(email);
           customerHelper.setPassword(password);
         }
        shoppingList = new ArrayList();
}

 public void ejbCreate(CustomerHelper customerHelper)
    throws CreateException {
        customerHelper.setName(customerHelper.name);
        customerHelper.setEmail(customerHelper.email);
        customerHelper.setPassword(customerHelper.password);
        shoppingList = new ArrayList();
}
 // Business methods implementation

public CustomerHelper getCustomer() {
    return customerHelper;
}
```

ビジネス・オブジェクトのサンプル

次のサンプル画面は、 39 ページの『EJB JAR ファイル・コード』に示されているソース・コードに対応するビジネス・オブジェクト構造を示しています。このビジネス・オブジェクトは ODA によって作成されます。これは、元の EJB JAR ファイルで定義されたオブジェクトと構成体をディスカバーし、対応するビジネス・オブジェクトを生成します。ODA を使用してこの例を生成する方法については、 43 ページの『第 5 章 ビジネス・オブジェクトの作成と変更』を参照してください。

4 番目の属性 getCustomer に注意してください。この属性には、 39 ページの『EJB JAR ファイル・コード』に示されている MusicCartBean クラスのサンプル・コードの最後に定義されている getCustomer メソッドに対応する複合メソッド・オブジェクトが含まれています。

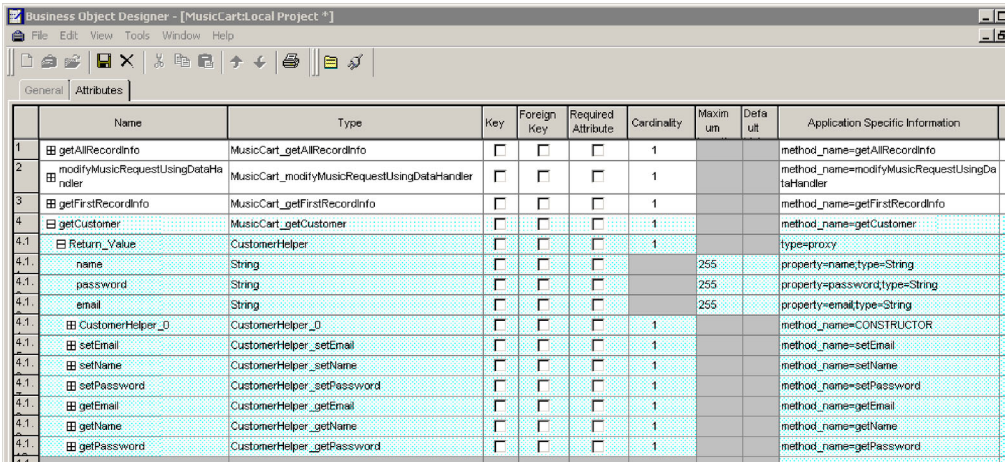


図 4. ビジネス・オブジェクト・レベル ASI とサポートされる動詞

ビジネス・オブジェクトの生成

実行時にイベントが発生するたびに、EJB アプリケーションは、オブジェクト・レベルのデータと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、対応するビジネス・オブジェクト定義にこのデータをマップし、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信します。また、コネクターは統合ブローカーから戻されたビジネス・オブジェクトを受信し、それを EJB アプリケーションに渡します。

注: EJB アプリケーション内のオブジェクト・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 EJB アプリケーションから送信されたデータと正確に一致しない場合は、コネクターでビジネス・オブジェクトを作成することができず、トランザクションは失敗します。

Business Object Designer には、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。詳細については、43 ページの『第 5 章 ビジネス・オブジェクトの作成と変更』を参照してください。

第 5 章 ビジネス・オブジェクトの作成と変更

この章では、EJB 用 ODA (Object Discovery Agent) を使用してビジネス・オブジェクトを作成する方法について説明します。この章の内容は次のとおりです。

EJB 用の ODA の概要

ODA を使用すると、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・オブジェクトを調べ、ビジネス・オブジェクト属性に対応するオブジェクトのエレメントを「ディスカバー」し、情報を表すビジネス・オブジェクト定義を生成します。Business Object Designer には、Object Discovery Agent にアクセスし、それを対話式に操作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。

EJB 用の ODA は、EJB JAR ファイルに含まれるメタデータからビジネス・オブジェクト定義を生成します。Business Object Designer 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。ビジネス・オブジェクトの作成には ODA を使用し、ビジネス・オブジェクトをサポートするようコネクターを構成するためには Connector Configurator を使用します。Connector Configurator の詳細については、77 ページの『付録 B. Connector Configurator』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 で EJB 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について説明します。Business Object Designer の起動と使用については、「*IBM WebSphere Business Integration Adapters 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

ODA は、メタデータ・リポジトリー (JAR ファイル) が常駐するファイル・システムをマウントできる任意のマシンから、ODA 始動ファイルを使用して実行できます。このファイルには、特定の必須 EJB およびコネクター JAR ファイルへのパスを含む始動パラメーターが格納されています。これらの JAR ファイルは、ODA を実行しているマシンからもアクセスできる必要があります。

EJB 用の ODA のデフォルト名は EJBODA です。この名前は、始動スクリプトで `AGENTNAME` 変数の値を変え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_EJBODA.bat` (Windows 2000)
- `start_EJBODA.sh` (Unix)

始動ファイルの CLIENT JARPATH 変数にはデフォルト値が含まれていないので、start コマンドを実行する前に、この値をクライアント JAR パスのディレクトリーおよびファイル名に設定してください。そうすれば ODA はこのファイルを Java クラス・パスに組み込むことができます。例えば、次のように指定します。

```
set CLIENTJARPATH=C:¥BIA_MusicBeanSample.jar
```

Business Object Designer の実行

Business Object Designer には、ODA を使用してビジネス・オブジェクト定義を生成する手順を示すウィザードが用意されています。ステップは次のとおりです。

エージェントの選択

エージェントを選択するには、次の手順を実行します。

1. Business Object Designer を始動します。
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 を選択し、**「次へ」**をクリックします。(希望の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」**をクリックしてください。)

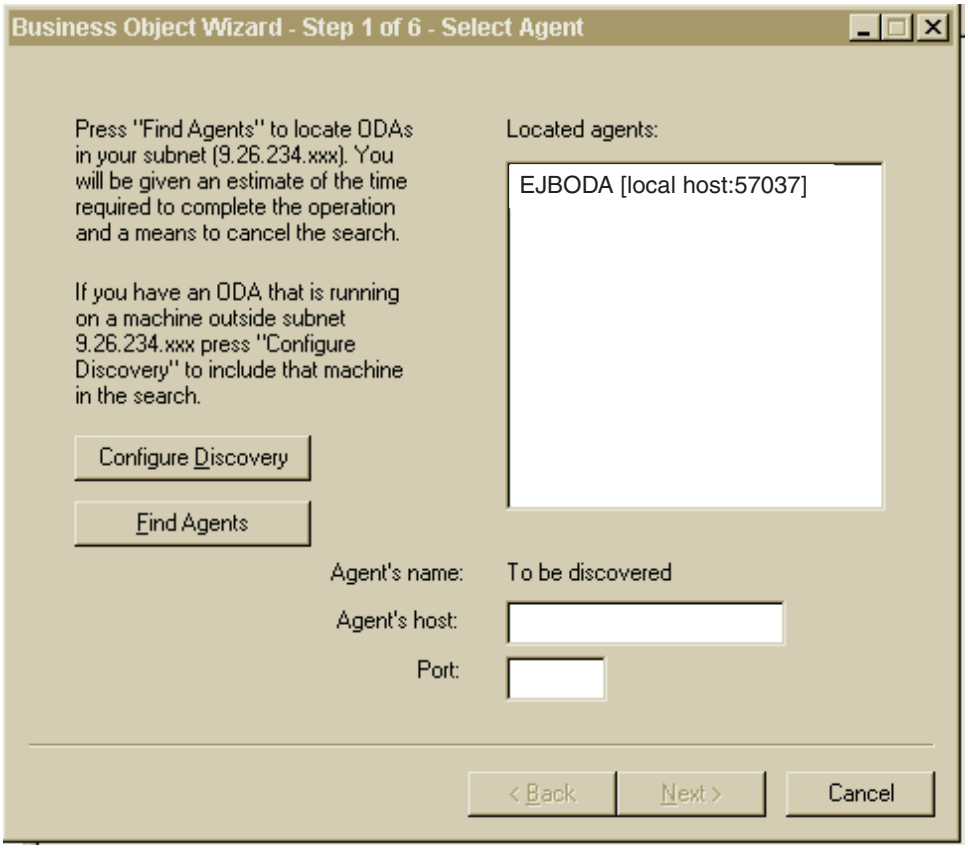


図 5. 「エージェントの選択」画面

エージェントの構成

「エージェントの選択」画面で**「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。図 6 に、この画面のサンプル値を示します。

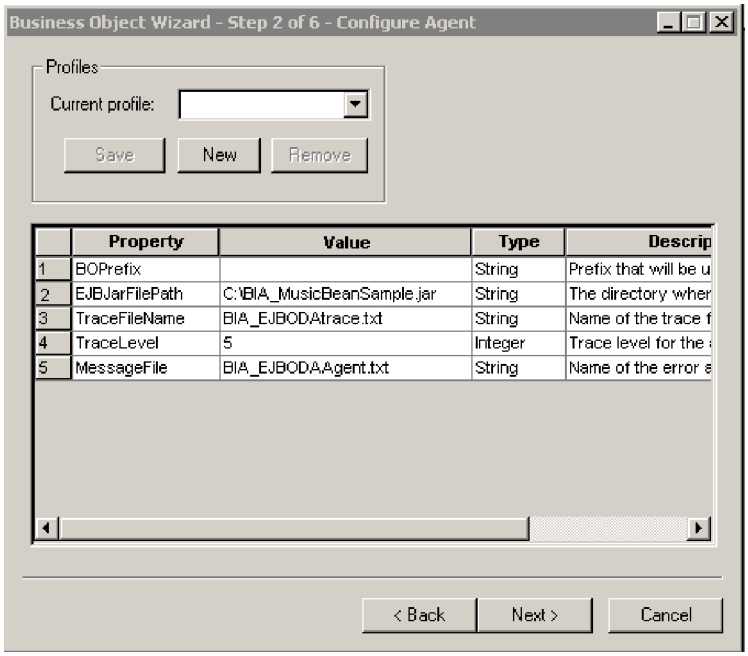


図 6. 「エージェントの構成」画面

表 9 で、この画面で設定するプロパティーについて説明します。この画面で入力したすべての値をプロファイルに保管できます。次に ODA を実行する際は、プロパティーを再入力するのではなく、「現在のプロファイル」ドロップダウン・メニューからプロファイルを選択して、保管した値を再利用します。プロファイルごとに異なる値を指定して、複数のプロファイルを保管することもできます。

表 9. 「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	Type	説明
EJBJarFilePath	なし	String	(必須) EJB JAR ファイルへの完全修飾パス。これは、BO がデータを交換する必要のある Enterprise Bean の定義を含むファイルです。
DefaultBOPrefix	なし	String	ODA が、生成したビジネス・オブジェクトの名前に追加するプレフィックス。
TraceFileName	<*agentname*>trace.txt	String	トレース・メッセージ・ファイルの名前。例えば、BIA_EJBODAtrace.txt。コネクターは、このプロパティーに関してはグローバル化された値をサポートします。つまり、ファイル・パスおよびファイル名はグローバル化できます。ただし、ファイル内容はグローバル化できません。

表 9. 「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	Type	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、 57 ページの『トレース』を参照してください。
MessageFile	*<agentname>*Agent.txt	String	ODA によって表示されるメッセージをすべて含むメッセージ・ファイルの名前。Connector for EJB の場合、このファイルの名前は BIA_EJBODAAgent.txt です。メッセージ・ファイルの名前を正しく指定しないと、ODA はメッセージなしで動作します。コネクターは、このプロパティーに関してはグローバル化された値をサポートします。つまり、ファイル・パスおよびファイル名はグローバル化できます。ただし、ファイル内容はグローバル化できません。

1. 初めて ODA を実行して新規プロファイルを作成する場合は、「プロファイル」グループ・ボックスの**「新規」**ボタンと**「保管」**ボタンを使用します。ODA を再び使用する場合は、既存プロファイルを選択します。
2. 45 ページの表 9 の説明に従って、各プロパティーの名前、値、タイプ、および説明を入力します。

   注: プロファイルを使用する場合、プロパティー値はあらかじめ入力されていますが、必要に応じてその値を変更できます。

ビジネス・オブジェクトの選択

47 ページの図 7 に示されている*「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。画面には、EJB JAR ファイルで定義されたオブジェクトのリストが表示されます。この画面を使用して、ODA でビジネス・オブジェクト定義を生成する EJB オブジェクトを必要な数だけ選択します。

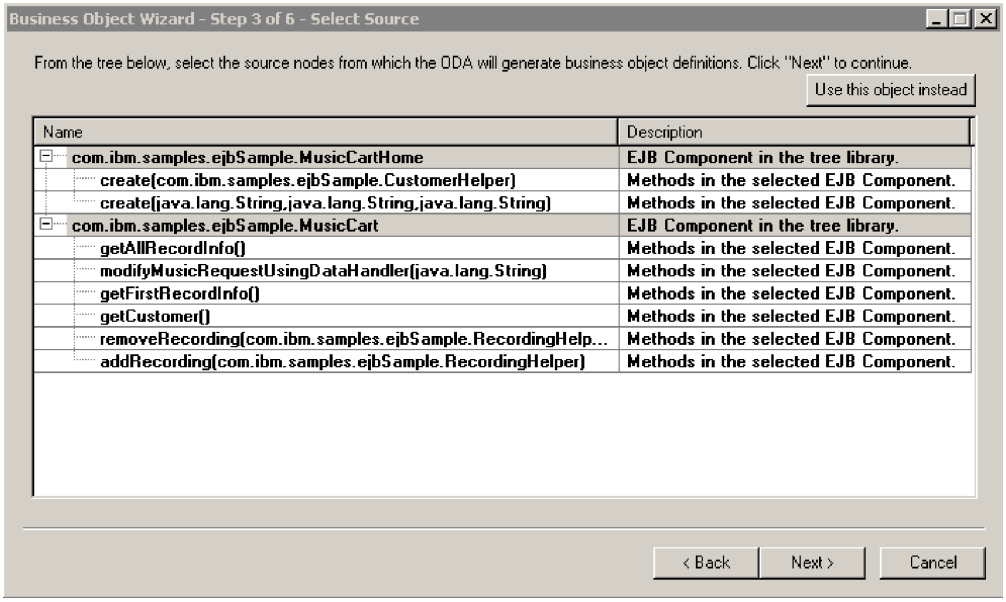


図 7. 「ソースの選択」画面

1. 必要に応じて EJB オブジェクトを展開し、オブジェクトのメソッドのリストを参照します。
2. 使用する EJB オブジェクトを選択します。図 7 では、`MusicCartHome` クラスと `MusicCart` クラスが選択されています。選択されているのはクラス名だけですが、ODA は、選択された各クラスの関連メソッドのオブジェクトも作成します。
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。

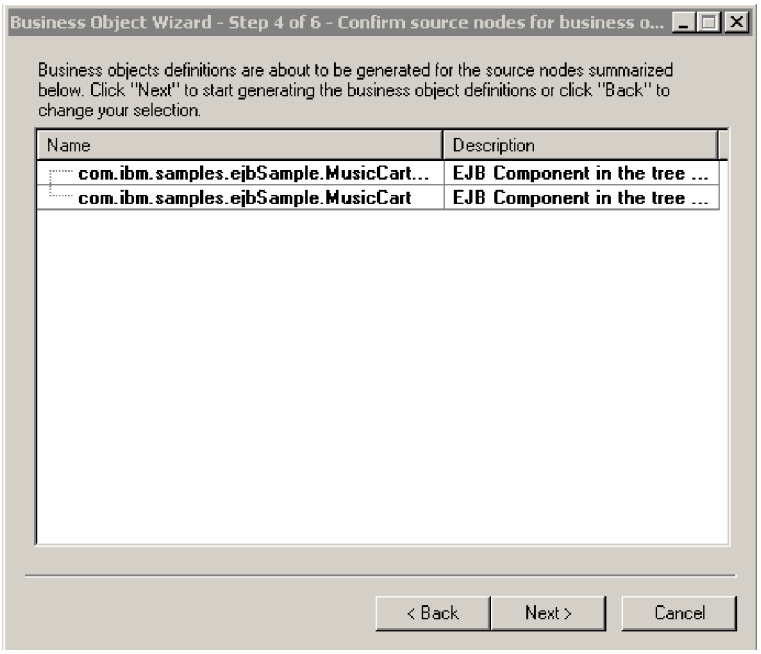


図 8. 「ソース・ノードの確認」画面

「戻る」をクリックして変更するか、**「次へ」**をクリックしてリストが正しいことを確認します。

*「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、ウィザードでビジネス・オブジェクトが生成中であることを示すメッセージが表示されます。

動詞の選択と JNDI 名の割り当て

「次へ」をクリックすると、*「この Enterprise Java Bean の JNDI 名を入力 (Enter JNDI name for this Enterprise Java Bean)」*画面が表示されます。この画面で、この

オブジェクトに対してサポートされる動詞を選択し、JNDI 名を割り当てます。図 9 の例では、割り当てられた JNDI 名は MusicCartJNDI です。

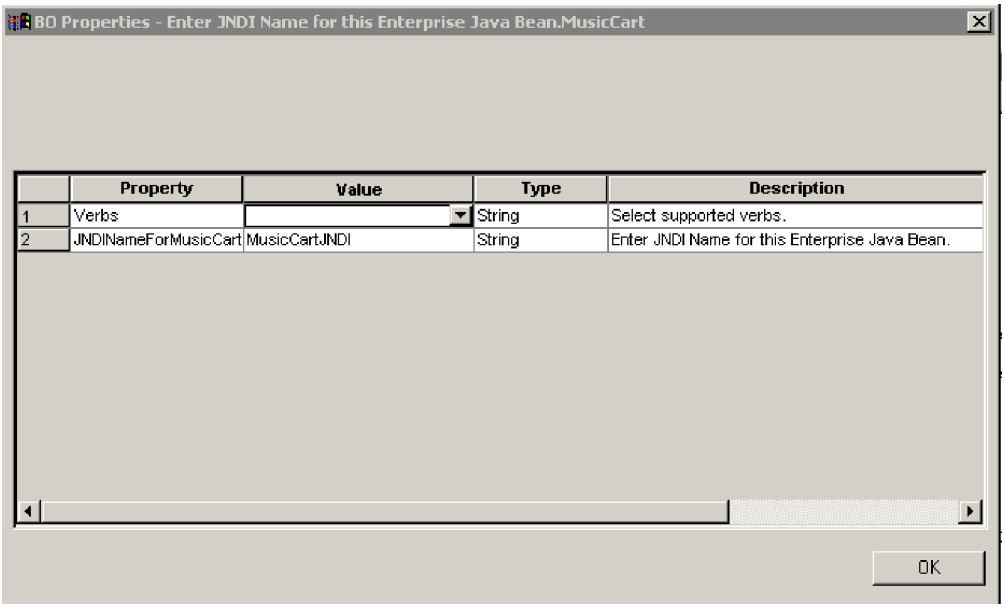
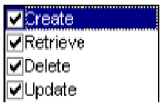


図 9. 「Enterprise Java Bean の JNDI 名を入力 (Enter JNDI name for Enterprise Java Bean)」画面

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も指定します。ODA では、サポートされている 4 つの動詞 (Create、Retrieve、Delete、および Update) を指定し、各動詞のアクションとして n 個のメソッドを割り当てることができます。ここで n は、対応する EJB オブジェクト内のメソッドの数と同じです。サポートされている 4 つの動詞以外の動詞を指定する場合や、ビジネス・オブジェクトの作成後に動詞情報を編集する場合は、Business Object Designer を使用します。

EJB コネクターのビジネス・オブジェクトの動詞の詳細については、32 ページの『動詞 ASI』を参照してください。

1. Verbs プロパティーの **「値」**リストで、ビジネス・オブジェクトでサポートする動詞を選択します。1 つ以上の動詞を選択できます。動詞はいつでも選択解除できます。



2. JNDI name for <*className*> プロパティーで、Enterprise Bean の JNDI 名を入力します。この名前は、Bean がアプリケーション・サーバーに配置されるときに定義されます。
3. **「OK」**をクリックします。

動詞 ASI の指定

選択した動詞ごとに、個別のウィンドウが表示されます。このウィンドウで、動詞に対して実行する必要のあるメソッド・シーケンスを指定します。

図 10 に、 47 ページの図 7 および 47 ページの図 8 で作成した MusicCart ビジネス・オブジェクトの Create 動詞に対するこの画面を示します。

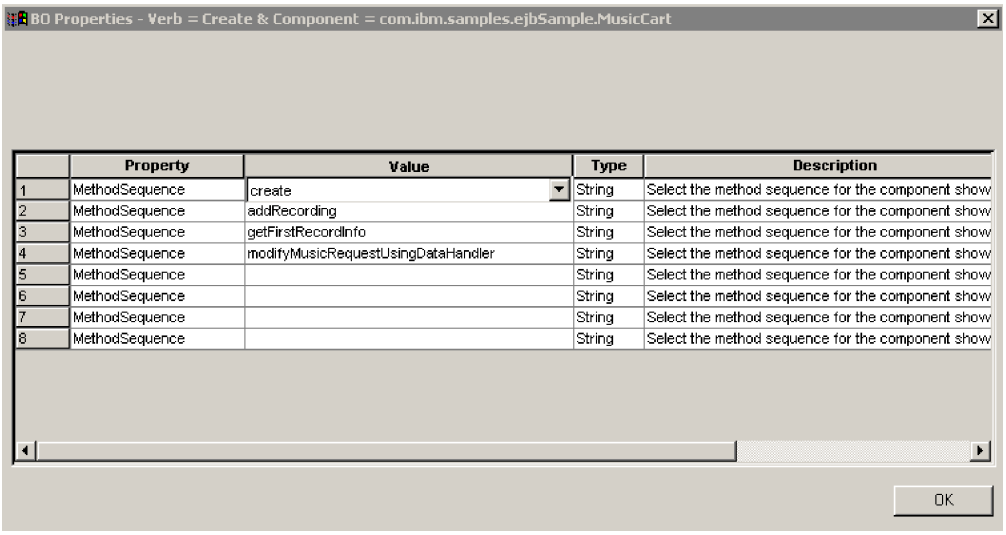


図 10. 動詞のメソッド・シーケンスの設定

1. MethodSequence プロパティーの**「値」**リストで、ビジネス・オブジェクトで動詞に対して最初に実行するメソッドをクリックします。図 10 では、メソッド・シーケンスが次のようになっています。
   - どの EJB ビジネス・オブジェクト動詞のメソッド・シーケンスでも最初に必要となるメソッドは creator または finder メソッドで、これらは Enterprise Bean のホーム・インターフェースに定義されています。図 10 では、最初のメソッドは create です。コネクターは、このメソッドを使用してリモート・インターフェースへの参照を取得し、Enterprise Bean のインスタンスを作成します。
   - シーケンス内の 2 番目のメソッドは addRecording です。このメソッドは、Enterprise Bean のリモート・インターフェースに属します。
   - シーケンス内の 3 番目のメソッドは getFirstRecordInfo です。このメソッドは、Enterprise Bean のリモート・インターフェースに属します。
   - シーケンス内の最後のメソッドは modifyMusicRequestUsingDataHandler です。このメソッドは、Enterprise Bean のリモート・インターフェースに属します。

   動詞のメソッド・シーケンスを指定することによって、その動詞に関連する動詞 ASI を作成します。必要な場合は、Business Object Designer を使用してこの動詞 ASI を後で変更することもできます。
2. **「OK」**をクリックします。

ビジネス・オブジェクトのオープンと保管

*「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」*画面が表示されます。

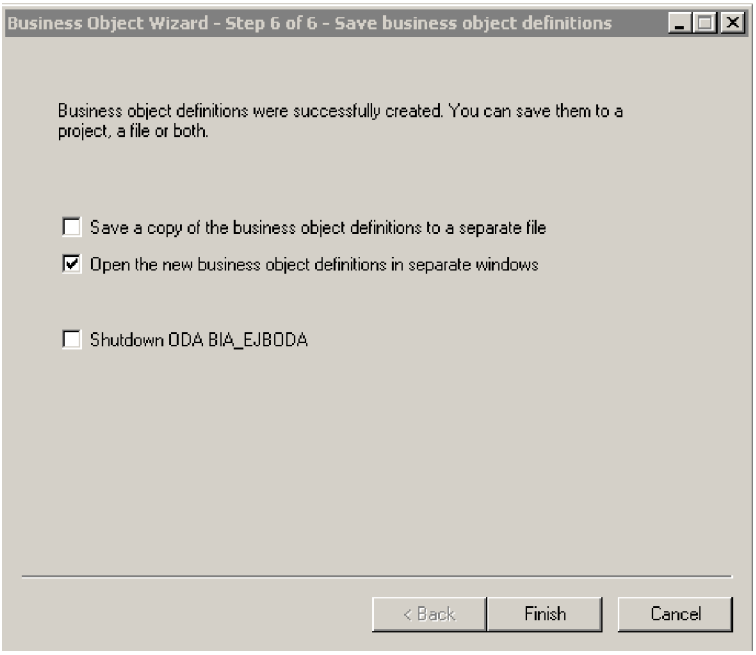


図 11. 「ビジネス・オブジェクトの保管」画面

オプションで、Business Object Designer 内の個別ウィンドウで新規ビジネス・オブジェクトを開いたり、(トップレベル・ビジネス・オブジェクトのキーを指定した後で) 生成されたビジネス・オブジェクト定義をファイルに保管したりできます。

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。
2. **「完了」**をクリックします。各ビジネス・オブジェクトが個別のウィンドウに表示されます。このウィンドウで、作成したビジネス・オブジェクトやビジネス・オブジェクト動詞の ASI 情報を設定できます。詳細については、47 ページの『動詞の選択と JNDI 名の割り当て』および 48 ページの『動詞 ASI の指定』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、次の手順を実行します (ただし、まず親レベルのビジネス・オブジェクトのキーを 52 ページの図 12 に示されているように指定する必要があります)。

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 によって、指定した場所にファイルが保管されます。

ODA での作業を完了したら、「ODA EJBODA のシャットダウン (Shutdown ODA EJBODA)」をオンにしてから**「完了」**をクリックして ODA をシャットダウンします。

属性レベルの ASI の指定

(各動詞に対して実行する必要のあるメソッド・シーケンスを指定することによって) 動詞 ASI を定義すると、Business Object Designer によってビジネス・オブジェクトの属性が表示されます。EJB コネクターでの属性レベル ASI の詳細については、34 ページの『属性レベル ASI』を参照してください。

各属性は、**「位置」**列の数値で定義されるビジネス・オブジェクト構造での表示順で、**「属性」**タブに表示されます。単純 EJB オブジェクト属性は単純属性として表示され、その ASI には元の EJB 属性の名前とタイプが含まれます。

画面には、各属性の名前、タイプ、および ASI 情報が示されます。52 ページの図 12 に、メソッド (複合) 属性 ASI を示します。ビジネス・オブジェクトの getAllRecordsInfo 属性には、属性を元の EJB オブジェクト・メソッドにマップする ASI があります。この例では、元のメソッドは、method_name=getAllRecordInfo ASI によって**「アプリケーション固有情報」**列の下に示されています。

ビジネス・オブジェクトの getCustomer 属性には、属性を元の EJB オブジェクト・メソッドにマップする ASI があります。この例では、元のメソッドは、method_name=getCustomer ASI によって**「アプリケーション固有情報」**列の下に示されています。

また、getCustomer (子ビジネス・オブジェクト) には、子オブジェクト属性 Return_Value があります。これは、getCustomer メソッドの戻り値を取り込むために使用されます。EJB JAR ファイルでは、このメソッドは、それ自体がメソッドであるタイプ CustomerHelper の戻り値をとるよう定義されています。オブジェクト・タイプが戻り値であるため、CustomerHelper の属性 ASI は type=proxy に設定されます。EJB JAR ファイル内のメソッドが値を戻さない場合、Return_Value 属性は、ビジネス・オブジェクト属性のリストに組み込まれません。

また、タイプ CustomerHelper の Return_Value 属性には、name、password、および email の、3 つの非メソッド子オブジェクト属性があります。MusicCartBean クラスが定義されている元の EJB JAR ファイルでは、これらはすべて CustomerHelper メソッドのタイプ String のパラメーターとして定義されています。ビジネス・オブジェクトでは、これらのパラメーターの属性 ASI が、元の JAR ファイルにおけるそれぞれの Java プロパティー名とタイプ (String) を示します。

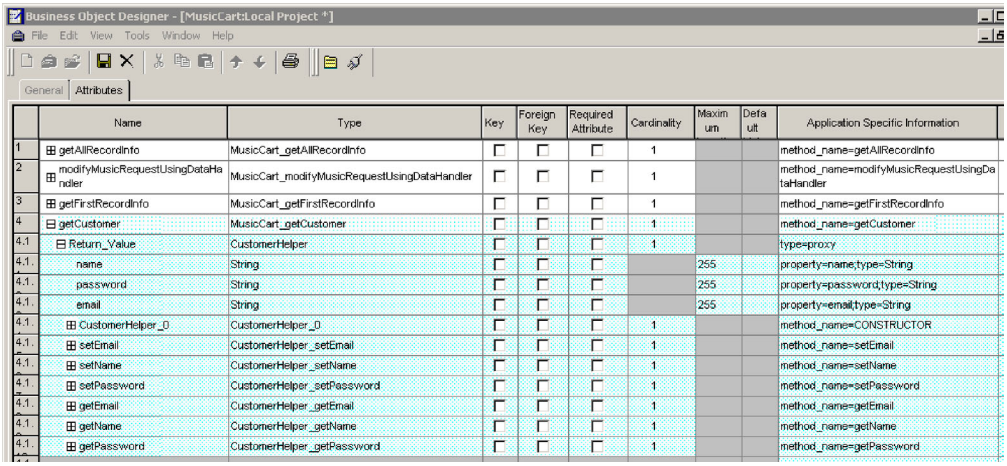


図 12. 属性 ASI の設定

この画面では、親レベルのオブジェクトがキーであるかどうかを指定します (これは、ODA でビジネス・オブジェクトを個別のファイルに保管するために必要です)。また、この画面では、必要に応じて子オブジェクト・キーを設定し、次の情報を指定します。

- コネクターがビジネス・オブジェクトを処理するためにこの属性が必要かどうか。必要である場合は、**「必要」**チェック・ボックスをオンにします。
- 属性の最大長が、**「最大長」**列に表示されている値と異なっているかどうか。
- 属性にデフォルト値があるかどうか。ある場合は、**「デフォルト」**列に値を入力します。

注: ビジネス・オブジェクトを ODA (Business Object Designer で実行) で作成して親レベルのキーを設定することはできますが、外部キーをこの方法で構成することはできません。外部キーは、非 ASI メタデータなので、必ず ODA を使用せずに構成する必要があります (Business Object Designer で、**「ファイル」**> **「新規」**をクリックし、ODA を使用せずに新規ビジネス・オブジェクトを作成します)。

ビジネス・オブジェクト・レベル ASI の指定

属性レベル ASI の指定後、ビジネス・オブジェクト・レベル ASI を表示および変更することができます。ビジネス・オブジェクト・レベル ASI の詳細については、31 ページの『ビジネス・オブジェクト・レベル ASI』を参照してください。

ビジネス・オブジェクト・レベル ASI は、**「一般」**タブにリストされています。**「ビジネス・オブジェクト・レベル・アプリケーション固有の情報」**フィールドに表示される ASI 値には、このビジネス・オブジェクトに対応するリモート・インターフェースの名前が含まれています。コネクターは、この情報を使用してリモート・オブジェクトをビジネス・オブジェクトにマップします。

また、**「一般」**タブには、ビジネス・オブジェクトによってサポートされる動詞がすべてリストされ、 48 ページの『動詞 ASI の指定』で定義されているように、各動詞の ASI が示されます。動詞がブランクである場合、その動詞に対してメソッド・シーケンスは実行されません。

ビジネス・オブジェクト・ファイルのアップロード

新規作成したビジネス・オブジェクト定義ファイルは、作成後すぐに統合ブローカーにアップロードする必要があります。プロセスは、WebSphere InterChange Server、WebSphere MQ Integrator Broker、または WebSphere Application Server のいずれを実行しているかによって異なります。

- **WebSphere InterChange Server**: ビジネス・オブジェクト定義ファイルがローカル・マシンに保管されていて、それ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、InterChange Server のインプリメンテーション資料を参照してください。
- **WebSphere MQ Integrator Broker**: ビジネス・オブジェクト定義を Business Object Designer からエクスポートし、統合ブローカーにインポートする必要があります。詳細については、WebSphere MQ Integrator Broker のインプリメンテーション資料を参照してください。
- **WebSphere Application Server**: 詳細については、WebSphere Application Server のインプリメンテーション資料を参照してください。

第 6 章 トラブルシューティングとエラー処理

この章では、Connector for EJB がエラーを処理する方法について説明します。このコネクターは、ロギング・メッセージとトレース・メッセージを生成します。この章では、これらのメッセージについて説明し、トラブルシューティングのヒントを示します。この章の内容は次のとおりです。

エラー処理

コネクターが生成するメッセージはすべて、BIA_EJBConnector.txt という名前のメッセージ・ファイルに保管されます。(ファイル名は、LogFileName 標準コネクター構成プロパティーによって決定されます。)各メッセージにはメッセージ番号があり、その後にメッセージが続きます。

```
Message number
Message text
```

コネクターは、以下の各セクションで説明するような特定のエラーを処理します。

ClassNotFound for proxy

この例外エラーは、コネクターとのデータ交換のためユーザーによって提供された EJB JAR ファイル内で、コネクターが所定のリモート・インターフェースを検出できない場合に発生します。コネクターは、検出されなかったクラスの名前を含むエラーをログに記録し、FAIL コードを戻します。

InstantiationException in Loader

この例外エラーは、コネクターが Enterprise Bean クラス名を受信してそのクラスのオブジェクトを作成しようとしたときに、オブジェクト・インスタンスを生成できなかった場合に発生します。コネクターは、インスタンスを生成できなかったオブジェクトのクラス名を含むエラーをログに記録し、FAIL コードを戻します。

Illegal AccessException in Loader or Invoker

メソッドに無効コードまたは不適切なアクセス (パブリックまたはプライベート) がある場合、コネクターは例外エラーを生成します。

コネクターはエラーをログに記録し、FAIL コードを戻します。

NoSuchMethodException in Invoker

対応する Enterprise Bean オブジェクトに存在しないメソッドがビジネス・オブジェクトで指定されている場合、コネクターは例外エラーを生成します。メソッドは動的にロードされるので、クラス内にメソッドが検出されない場合、この例外エラーが生成されます。

コネクターはエラーをログに記録し、FAIL コードを戻します。

InvocationTargetException in Invoker

(コネクターがビジネス・オブジェクトを交換している) EJB アプリケーションで例外エラーが生成された場合、コネクターは例外エラーを生成します。

コネクターはエラーをログに記録し、FAIL コードを戻します。

Invalid argument (CXIgnore) in a method object in Invoker

ビジネス・オブジェクトの動詞 ASI にメソッドが組み込まれている場合に、そのメソッドの引き数にデータが組み込まれていないと、コネクターは例外エラーを生成します。

コネクターはエラーをログに記録し、FAIL コードを戻します。

Cast failure or wrong attribute type

EJB オブジェクト・メソッドが、ビジネス・オブジェクトで指定されたものと異なるデータ型をとった場合や戻した場合、コネクターは例外エラーを生成します。

コネクターはエラーをログに記録し、FAIL コードを戻します。

Invalid verb ASI

コネクターに渡されるビジネス・オブジェクトの動詞 ASI で、その形式が正しくない場合や不適切な構文が使用されている場合、コネクターは例外エラーを生成します。この例として、適切なメソッド・シーケンスを含んでいない動詞 ASI があります。

コネクターはエラーをログに記録し、FAIL コードを戻します。

App response timeout

Enterprise Bean が配置されているアプリケーション・サーバーへの接続が失われた場合や、接続の失敗のため EJB メソッドを起動できない場合、コネクターは例外エラーを生成します。

コネクターはエラーをログに記録し、FAIL コードを戻します。

ロギング

55 ページの『エラー処理』に記載されているエラーはすべてメッセージ・ファイル (BIA_EJBConnector.txt) から読み取られます。

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、15 ページの『コネクターの構成』のコネクター構成プロパティーを参照してください。トレースの有効化および設定の方法など、詳細については、「*コネクター開発ガイド*」を参照してください。

表 10 に、推奨されるコネクター・トレース・メッセージ・レベルの内容を示します。

表 10. 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、その他のトレースは実行されません。
レベル 1	このレベルは、次のトレース・メッセージに使用します。 • 状況情報を提供するトレース・メッセージ。 • 処理される各ビジネス・オブジェクトに関するキー情報を提供するトレース・メッセージ。 • ポーリング・スレッドが入力キューで新規メッセージを検出するたびに記録するトレース・メッセージ。
レベル 2	このレベルは、次のトレース・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 BO ハンドラーを識別するトレース・メッセージ。 • ビジネス・オブジェクトが統合ブローカーに通知されるたびにログを記録するトレース・メッセージ。 • 要求ビジネス・オブジェクトが受信されるたびにそれを示すトレース・メッセージ。
レベル 3	このレベルは、次のトレース・メッセージに使用します。 • 該当する場合に、処理される外部キーを識別するトレース・メッセージ。これらのメッセージは、コネクターがビジネス・オブジェクト内で外部キーを検出した場合やコネクターがビジネス・オブジェクト内で外部キーを設定した場合に表示されます。 • ビジネス・オブジェクト処理に関連するトレース・メッセージ。この例としては、ビジネス・オブジェクト間での一致の検索や、子ビジネス・オブジェクトの配列でのビジネス・オブジェクトの検索があります。
レベル 4	このレベルは、次のトレース・メッセージに使用します。 • アプリケーション固有情報を識別するトレース・メッセージ。この例としては、ビジネス・オブジェクトでアプリケーション固有情報を処理するメソッドによって戻される値があります。 • コネクターがある機能を開始または終了した時点を識別するトレース・メッセージ。これらのメッセージは、コネクターの処理フローのトレースに役立ちます。 • スレッド固有の処理を記録するトレース・メッセージ。例えば、コネクターが複数のスレッドを作成する場合、メッセージは各新規スレッドの作成をログに記録します。

表 10. トレース・メッセージの内容 (続き)

レベル	説明
レベル 5	このレベルは、次のトレース・メッセージに使用します。 • コネクターの初期化を示すトレース・メッセージ。このタイプのメッセージには、例えば、ブローカーから検索された各 Connector Configurator プロパティーの値があります。 • コネクターが実行中に作成した各スレッドの状況を詳細に記録するトレース・メッセージ。 • アプリケーション内で実行されるステートメントを示すトレース・メッセージ。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) が入ります。 • ビジネス・オブジェクト・ダンプを記録するトレース・メッセージ。コネクターは、ビジネス・オブジェクトの処理を開始する前とビジネス・オブジェクトを処理した後に、そのオブジェクトのテキスト表現 (処理前にはコネクターがコラボレーションから受信したオブジェクトを示すもの、処理後にはコネクターがコラボレーションに戻したオブジェクトを示すもの) を出力する必要があります。

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 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。この付録の内容は、次の統合ブローカーで実行されるコネクターを対象としています。

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator、WebSphere MQ Integrator Broker、および WebSphere Business Integration Message Broker。これらをまとめて WebSphere Message Brokers (WMQI) と呼びます。
- WebSphere Application Server (WAS)

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

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

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

新規プロパティーと削除されたプロパティー

本リリースには、次の標準プロパティーが追加されました。

新規プロパティー

- XMLNameSpaceFormat

削除されたプロパティー

- RestartCount

標準コネクター・プロパティーの構成

アダプター・コネクターには 2 つのタイプの構成プロパティーがあります。

- 標準構成プロパティー
- コネクター固有のプロパティー

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

Connector Configurator の使用

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

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

プロパティー値の設定と更新

プロパティー・フィールドのデフォルトの長さは 255 文字です。

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

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

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

- **動的**
  変更を System Manager に保管すると、変更が即時に有効になります。例えば WebSphere Message Brokers で稼働している場合など、コネクターがスタンドアロン・モードで (System Manager から独立して) 稼働している場合は、構成ファイルでのみプロパティーを変更できます。この場合、動的更新は実行できません。
- コンポーネント再始動
  System Manager でコネクターを停止してから再始動しなければ、変更が有効になりません。アプリケーション固有コンポーネントまたは統合ブローカーを停止、再始動する必要はありません。
- サーバー再始動
  アプリケーション固有のコンポーネントおよび統合ブローカーを停止して再始動しなければ、変更が有効になりません。
- エージェント再始動 (ICS のみ)
  アプリケーション固有のコンポーネントを停止して再始動しなければ、変更が有効になりません。

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

標準プロパティーの要約

表 11 は、標準コネクター構成プロパティーの早見表です。標準プロパティーの依存関係は RepositoryDirectory に基づいているため、コネクターによっては使用されないプロパティーがあり、使用する統合ブローカーによってプロパティーの設定が異なる可能性があります。

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

表 11. 標準構成プロパティーの要約

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

表 11. 標準構成プロパティーの要約 (続き)

プロパティー名	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	JMS トランスポートのみ、Container Managed Events は <NONE> でなければならない
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory または CxCommon.Messaging.jms.SonicMQFactory または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	FactoryClassName が IBM の場合は crossworlds.queue.manager を使用。FactoryClassName が Sonic の場合 localhost:2506 を使用。	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMaxNativeStackSize	スタックのサイズ (キロバイト単位)	128k	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	Repository Directory は <REMOTE>
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならない
Locale	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR **注:** これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	True または False	False	コンポーネント再始動	Repository Directory は <REMOTE> でなければならない

表 11. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならない
MessageFileName	パスまたはファイル名	InterchangeSystem.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	*CONNECTORNAME*/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は True でなければならない
OADAutoRestartAgent	True または False	False	動的	Repository Directory は <REMOTE> でなければならない
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければならない
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければならない
PollEndTime	HH:MM	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒) no (ポーリングを使用不可にする) key (コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力された場合にのみポーリングする)	10000	動的	
PollQuantity	1 から 500	1	エージェント再始動	JMS トランスポートのみ: Container Managed Events を指定
PollStartTime	HH:MM (HH は 0 から 23、MM は 0 から 59)	HH:MM	コンポーネント再始動	
RepositoryDirectory	メタデータ・リポジトリーの場所		エージェント再始動	ICS の場合は <REMOTE> に設定する。WebSphere MQ Message Brokers および WAS の場合: C:¥crossworlds¥repository に設定する

表 11. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
RequestQueue	有効な JMS キュー名	*CONNECTORNAME*/REQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
ResponseQueue	有効な JMS キュー名	*CONNECTORNAME*/RESPONSEQUEUE	コンポーネント再始動	Delivery Transport が JMS の場合: Repository Directory が <REMOTE> の場合のみ必要
RestartRetryCount	0 から 99	3	動的	
RestartRetryInterval	適切な正数 (単位: 分) : 1 から 2147483547	1	動的	
RHF2MessageDomain	mrm、xml	mrm	コンポーネント再始動	Delivery Transport が JMS で、かつ WireFormat が CwXML の場合のみ
SourceQueue	有効な WebSphere MQ 名	*CONNECTORNAME*/SOURCEQUEUE	エージェント再始動	Delivery Transport が JMS であり、かつ Container Managed Events が指定されている場合のみ
SynchronousRequestQueue		*CONNECTORNAME*/ SYNCHRONOUSREQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
SynchronousRequestTimeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	Delivery Transport は JMS
SynchronousResponseQueue		*CONNECTORNAME*/ SYNCHRONOUSRESPONSEQUEUE	コンポーネント再始動	Delivery Transport は JMS
WireFormat	CwXML、CwBO	CwXML	エージェント再始動	Repository Directory が <REMOTE> でない場合は Repository Directory が <REMOTE> の場合は CwBO
WsifSynchronousRequest Timeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	WAS のみ
XMLNameSpaceFormat	short、long	short	エージェント再始動	WebSphere MQ Message Brokers および WAS のみ

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

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

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

RepositoryDirectory が <REMOTE> の場合のみ適用されます。

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

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

AgentTraceLevel

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

ApplicationName

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

BrokerType

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

CharacterEncoding

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

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

デフォルトでは、ドロップ・リストには、サポートされる文字エンコードの一部のみが表示されます。ドロップ・リストに、サポートされる他の値を追加するには、

製品ディレクトリーにある ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。詳細については、Connector Configurator に関する付録を参照してください。

ConcurrentEventTriggeredFlows

RepositoryDirectory が <REMOTE> の場合のみ適用されます。

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

このプロパティーを 1 よりも大きい値に設定すると、ソース・アプリケーションのコネクターが、複数のイベント・ビジネス・オブジェクトを同時にマップして、複数のコラボレーション・インスタンスにそれらのビジネス・オブジェクトを同時にデリバリーすることができます。これにより、統合ブローカーへのビジネス・オブジェクトのデリバリーにかかる時間、特にビジネス・オブジェクトが複雑なマップを使用している場合のデリバリー時間が短縮されます。ビジネス・オブジェクトのコラボレーションに到達する速度を増大させると、システム全体のパフォーマンスを向上させることができます。

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

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

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

ContainerManagedEvents

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

デフォルト値は No value です。

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

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

また、MimeType、DHClass、および DataHandlerConfigMOName (オプショナル) プロパティーを設定したデータ・ハンドラーも構成する必要があります。これらのプロパティーの値を設定するには、Connector Configurator の**「データ・ハンドラー」**タブを使用します。「データ・ハンドラー」タブの値のフィールドは、ContainerManagedEvents を JMS に設定した場合にのみ表示されます。

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

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

ControllerStoreAndForwardMode

RepositoryDirectory が <REMOTE> の場合のみ適用されます。

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

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

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

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

デフォルト値は true です。

ControllerTraceLevel

RepositoryDirectory が <REMOTE> の場合のみ適用されます。

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

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

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

- ICS がブローカー・タイプの場合は、DeliveryTransport プロパティーの指定可能な値は MQ、IDL、または JMS であり、デフォルトは IDL になります。
- RepositoryDirectory がローカル・ディレクトリーの場合は、指定可能な値は JMS のみです。

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

WebSphere MQ および IDL

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

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

JMS

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

JMS をデリバリー・トランスポートとして選択した場合は、
jms.MessageBrokerName、jms.FactoryClassName、jms.Password、jms.UserName などの追加の JMS プロパティーが Connector Configurator 内に表示されます。このうち最初の 2 つは、このトランスポートの必須プロパティーです。

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

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

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

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

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

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

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

DuplicateEventElimination

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

このプロパティーは、false に設定することもできます。

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

FaultQueue

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

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

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

デフォルト値は 128M です。

JvmMaxNativeStackSize

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

デフォルト値は 128K です。

JvmMinHeapSize

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

デフォルト値は 1M です。

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

デフォルト値は crossworlds.queue.manager です。

jms.NumConcurrentRequests

コネクターに対して同時に送信することができる並行サービス呼び出し要求の数 (最大値) を指定します。この最大値に達した場合、新規のサービス呼び出し要求はブロックされ、既存のいずれかの要求が完了した後で処理されます。

デフォルト値は 10 です。

jms.Password

JMS プロバイダーのためのパスワードを指定します。このプロパティーの値はオプションです。

デフォルトはありません。

jms.UserName

JMS プロバイダーのためのユーザー名を指定します。このプロパティーの値はオプションです。

デフォルトはありません。

ListenerConcurrency

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

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

Locale

言語コード、国または地域、および、希望する場合には、関連した文字コード・セットを指定します。このプロパティーの値は、データの照合やソート順、日付と時刻の形式、通貨記号などの国/地域別情報を決定します。

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、以下のように説明されます。

ll	2 文字の言語コード (普通は小文字)
TT	2 文字の国または地域コード (普通は大文字)
codeset	関連文字コード・セットの名前。名前のこの部分は、通常、オプションです。

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

デフォルト値は en_US です。コネクターがグローバル化に対応していない場合、このプロパティーの有効な値は en_US のみです。特定のコネクターがグローバル化に対応しているかどうかを判別するには、以下の Web サイトにあるコネクターのバージョン・リストを参照してください。

http://www.ibm.com/software/websphere/wbiadapters/infocenter、または
http://www.ibm.com/websphere/integration/wicserver/infocenter

LogAtInterchangeEnd

RespositoryDirectory が <REMOTE> の場合のみ適用されます。

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

例えば、LogAtInterChangeEnd を true に設定した場合にコネクターからアプリケーションへの接続が失われると、指定されたメッセージ宛先に、電子メール・メッセージが送信されます。デフォルト値は false です。

MaxEventCapacity

コントローラー・バッファー内のイベントの最大数。このプロパティーはフロー制御が使用し、RepositoryDirectory プロパティーの値が <REMOTE> の場合にのみ適用されます。

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

MessageFileName

コネクター・メッセージ・ファイルの名前です。メッセージ・ファイルの標準位置は ¥connectors¥messages です。メッセージ・ファイルが標準位置に格納されていない場合は、メッセージ・ファイル名を絶対パスで指定します。

コネクター・メッセージ・ファイルが存在しない場合は、コネクターは InterchangeSystem.txt をメッセージ・ファイルとして使用します。このファイルは、製品ディレクトリーに格納されています。

注: 特定のコネクターについて、コネクター独自のメッセージ・ファイルがあるかどうかを判別するには、該当するアダプターのユーザーズ・ガイドを参照してください。

MonitorQueue

コネクターが重複イベントをモニターするために使用する論理キューです。このプロパティーは、DeliveryTransport プロパティー値が JMS であり、かつ DuplicateEventElimination が TRUE に設定されている場合にのみ使用されます。

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

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

コネクターが自動再始動機能およびリモート再始動機能を使用するかどうかを指定します。この機能では、MQ により起動される Object Activation Daemon (OAD) を使用して、異常シャットダウン後にコネクターを再始動したり、System Monitor からリモート・コネクターを始動したりします。

自動再始動機能およびリモート再始動機能を使用可能にするには、このプロパティーを true に設定する必要があります。MQ により起動される OAD 機能の構成方法については、「*システム・インストール・ガイド (Windows 版)*」または「*システム・インストール・ガイド (UNIX 版)*」を参照してください。

デフォルト値は false です。

OADMaxNumRetry

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

異常シャットダウンの後で MQ により起動される OAD がコネクターの再始動を自動的に試行する回数の最大数を指定します。このプロパティーを有効にするには、OADAutoRestartAgent プロパティーを true に設定する必要があります。

デフォルト値は 1000 です。

OADRetryTimeInterval

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

MQ により起動される OAD の再試行時間間隔の分数を指定します。コネクター・エージェントがこの再試行時間間隔内に再始動しない場合は、コネクター・コントローラーはコネクター・エージェントを再び再始動するように OAD に要求します。OAD はこの再試行プロセスを OADMaxNumRetry プロパティーで指定された回数だけ繰り返します。このプロパティーを有効にするには、OADAutoRestartAgent プロパティーを true に設定する必要があります。

デフォルト値は 10 です。

PollEndTime

イベント・キューのポーリングを停止する時刻です。形式は HH:MM です。ここで、*HH* は 0 から 23 時を表し、*MM* は 0 から 59 分を表します。

このプロパティーには必ず有効な値を指定してください。デフォルト値は HH:MM ですが、この値は必ず変更する必要があります。

PollFrequency

ポーリング・アクション間の時間の長さです。PollFrequency は以下の値のいずれかに設定します。

- ポーリング・アクション間のミリ秒数。
- ワード key。コネクターは、コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力されたときにのみポーリングを実行します。このワードは小文字で入力します。
- ワード no。コネクターはポーリングを実行しません。このワードは小文字で入力します。

デフォルト値は 10000 です。

重要: 一部のコネクターでは、このプロパティーの使用が制限されています。このプロパティーが使用されるかどうかを特定のコネクターについて判別するには、該当するアダプター・ガイドのインストールと構成についての章を参照してください。

PollQuantity

コネクターがアプリケーションからポーリングする項目の数を指定します。アダプターにコネクター固有のポーリング数設定プロパティーがある場合、標準プロパティーの値は、このコネクター固有のプロパティーの設定値によりオーバーライドされます。

PollStartTime

イベント・キューのポーリングを開始する時刻です。形式は *HH:MM* です。ここで、*HH* は 0 から 23 時を表し、*MM* は 0 から 59 分を表します。

このプロパティーには必ず有効な値を指定してください。デフォルト値は HH:MM ですが、この値は必ず変更する必要があります。

RequestQueue

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

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

コネクターが XML スキーマ文書を読み取るリポジトリーの場所です。この XML スキーマ文書には、ビジネス・オブジェクト定義のメタデータが含まれています。

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

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

ResponseQueue

DeliveryTransport が JMS の場合のみ適用され、RepositoryDirectory が <REMOTE> の場合のみ必要です。

JMS 応答キューを指定します。JMS 応答キューは、応答メッセージをコネクター・フレームワークから統合ブローカーへデリバリーします。統合ブローカーが ICS の場合、サーバーは要求を送信し、JMS 応答キューの応答メッセージを待ちます。

RestartRetryCount

コネクターによるコネクター自体の再始動の試行回数を指定します。このプロパティーを並列コネクターに対して使用する場合、コネクターのマスター側のアプリケーション固有のコンポーネントがスレーブ側のアプリケーション固有のコンポーネントの再始動を試行する回数が指定されます。

デフォルト値は 3 です。

RestartRetryInterval

コネクターによるコネクター自体の再始動の試行間隔を分単位で指定します。このプロパティーを並列コネクターに対して使用する場合、コネクターのマスター側のアプリケーション固有のコンポーネントがスレーブ側のアプリケーション固有のコンポーネントの再始動を試行する間隔が指定されます。指定可能な値の範囲は 1 から 2147483647 です。

デフォルト値は 1 です。

RHF2MessageDomain

WebSphere Message Brokers および WAS のみ

このプロパティーを使用すると、JMS ヘッダーにあるフィールド・ドメイン名の値を構成できます。JMS トランスポートを介して WMQI にデータが送信されると、アダプター・フレームワークにより、JMS ヘッダー情報に加えて、ドメイン名および mrm の固定値が書き込まれます。構成可能なドメイン名を使用することにより、ユーザーは WMQI ブローカーによるメッセージ・データの処理状況を追跡できます。

サンプル・ヘッダーを次に示します。

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

デフォルト値は mrm ですが、xml に設定することもできます。このプロパティーが現れるのは、DeliveryTransport が JMS に設定され、WireFormat が CwXML に設定された場合のみです。

SourceQueue

DeliveryTransport が JMS で、ContainerManagedEvents が指定されている場合のみ適用されます。

JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、66 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

同期応答を要求する要求メッセージを、コネクター・フレームワークからブローカーに配信します。このキューは、コネクターが同期実行を使用する場合にのみ必要です。同期実行の場合、コネクター・フレームワークは、SynchronousRequestQueue にメッセージを送信し、SynchronousResponseQueue でブローカーから戻される応答を待機します。コネクターに送信される応答メッセージには、元のメッセージの ID を指定する 相関 ID が含まれています。

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

同期要求に対する応答として送信される応答メッセージを、ブローカーからコネクター・フレームワークに配信します。このキューは、コネクターが同期実行を使用する場合にのみ必要です。

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

コネクターが同期要求への応答を待機する時間を分単位で指定します。コネクターは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージをエラー・メッセージとともに障害キューに移動します。

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

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

WsifSynchronousRequest Timeout

WAS 統合ブローカーでのみ使用されます。

コネクターが同期要求への応答を待機する時間を分単位で指定します。コネクターは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージをエラー・メッセージとともに障害キューに移動します。

デフォルト値は 0 です。

XMLNameSpaceFormat

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

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

デフォルト値は short です。

付録 B. Connector Configurator

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

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

- コネクターを構成するためのコネクター固有のプロパティー・テンプレートを作成する
- 構成ファイルを作成する
- 構成ファイル内のプロパティーを設定する

注:

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

この付録では、次のトピックについて説明します。

Connector Configurator の概要

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

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator、WebSphere MQ Integrator Broker、および WebSphere Business Integration Message Broker。これらをまとめて WebSphere Message Brokers (WMQI) と呼びます。
- WebSphere Application Server (WAS)

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

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成します。
- **コネクター構成ファイル**を作成します。インストールするコネクターごとに構成ファイルを 1 つ作成する必要があります。
- 構成ファイル内のプロパティーを設定します。
  場合によっては、コネクター・テンプレートでプロパティーに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、ICS の場合はコラボレーションとともに使用するマップを

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

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

コネクター構成プロパティーには、標準の構成プロパティー (すべてのコネクターが持つプロパティー) 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により使用されるので、標準プロパティーを新規に定義する必要はありません。ファイルを作成すると、Connector Configurator により標準プロパティーがこの構成ファイルに挿入されます。ただし、Connector Configurator で各標準プロパティーの値を設定する必要があります。

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

ただし**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクター固有プロパティーのテンプレートを作成します。システム内で既にテンプレートが作成されている場合には、作成されているテンプレートを使用します。システム内でまだテンプレートが作成されていない場合には、79 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

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

Connector Configurator の始動

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

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

スタンドアロン・モードでの Configurator の実行

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

これを行うには、以下のステップを実行します。

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

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

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (84 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

1. 「System Manager」ウィンドウの「コネクター」フォルダーでいずれかの構成ファイルを選択し、右クリックします。Connector Configurator が開き、この構成ファイルの統合ブローカー・タイプおよびファイル名が上部に表示されます。
2. 「標準のプロパティー」タブをクリックし、この構成ファイルに含まれているプロパティーを確認します。

コネクター固有のプロパティー・テンプレートの作成

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

コネクター固有プロパティーのテンプレートを新規に作成するか、または既存のファイルをテンプレートとして使用します。

- テンプレートの新規作成については、『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でこのテンプレートを保管します。

新規テンプレートの作成

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

テンプレートは以下のように作成します。

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

     このテンプレートが使用されるコネクター (またはコネクターのタイプ) を表す固有の名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
   - **「旧テンプレート」、「変更する既存のテンプレートを選択してください」**

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

ご使用のコネクターで使用するコネクター固有のプロパティーが表示されるテンプレートが見つからない場合は、自分で作成する必要があります。

一般特性の指定

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

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

プロパティーの一般特性の選択を終えたら、**「値」**タブをクリックします。

値の指定

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

1. **「値」**タブをクリックします。「一般」のパネルに代わって「値」の表示パネルが表示されます。
2. **「プロパティーを編集」**表示でプロパティーの名前を選択します。
3. **「最大長」**および**「最大複数値」**のフィールドで、変更を行います。次のステップで説明するように、プロパティーの**「プロパティー値」**ダイアログ・ボックスを開かない限り、そのプロパティーの変更内容は受け入れられませんので、注意してください。
4. 値テーブルの左上の隅にあるボックスを右マウス・ボタンでクリックしてから、**「追加」**をクリックします。**「プロパティー値」**ダイアログ・ボックスが表示されます。このダイアログ・ボックスではプロパティーのタイプに応じて、値だけを入力できる場合と、値と範囲の両方を入力できる場合があります。適切な値または範囲を入力し、**「OK」**をクリックします。
5. **「値」**パネルが最新表示され、**「最大長」**および**「最大複数値」**で行った変更が表示されます。以下のような 3 つの列があるテーブルが表示されます。

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

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

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

   値が作成されて、グリッドに表示されると、そのテーブルの表示内から編集できるようになります。テーブルにある既存の値の変更を行うには、その行の行番号をクリックして行全体を選択します。次に**「値」**フィールドを右マウス・ボタンでクリックし、**「値の編集 (Edit Value)」**をクリックします。

依存関係の設定

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

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

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

   == (等しい)

   != (等しくない)

> (より大)

< (より小)

>= (より大か等しい)

<= (より小か等しい)

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

新しい構成ファイルを作成

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

ブローカーを選択するには、以下のステップを実行します。

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

また、以下の作業も実行できます。

- 「System Manager」ウィンドウで**「コネクター」**フォルダーを右クリックし、**「新規コネクターの作成」**を選択します。Connector Configurator が開き、**「新規コネクター」**ダイアログ・ボックスが表示されます。

コネクター固有のテンプレートからの構成ファイルの作成

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

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

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

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

- **システム接続**

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

- **「コネクター固有プロパティー・テンプレート」を選択します。**

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

  使用するテンプレートを選択し、**「OK」**をクリックします。

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

   重要: ここで設定するディレクトリー・パスおよび名前は、コネクターの始動ファイルで指定するコネクター構成ファイルのパスおよび名前に一致している必要があります。
5. この章で後述する手順に従って、「Connector Configurator」ウィンドウの各タブにあるフィールドに値を入力し、コネクター定義を完了します。

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

- コネクター定義ファイル。
  コネクター定義ファイルは、特定のコネクターのプロパティーと、適用可能なデフォルト値がリストされたテキスト・ファイルです。コネクターの配布パッケージの `¥repository` ディレクトリー内には、このようなファイルが格納されていることがあります (通常、このファイルの拡張子は `.txt` です。例えば、XML コネクターの場合は CN_XML.txt です)。
- ICS リポジトリー・ファイル。
  コネクターの以前の ICS インプリメンテーションで使用した定義は、そのコネクターの構成で使用されたリポジトリー・ファイルで使用可能になります。そのようなファイルの拡張子は、通常 `.in` または `.out` です。
- コネクターの以前の構成ファイル。
  これらのファイルの拡張子は、通常 *.cfg です。

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

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

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

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

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

     コネクターのアダプター・パッケージに *.txt ファイルが付属していた場合、または別の拡張子で定義ファイルが使用可能である場合は、このオプションを選択します。
3. ディレクトリー表示内で、適切なコネクター定義ファイルへ移動し、ファイルを選択し、**「開く」**をクリックします。

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

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

構成ファイルの完成

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

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

1. **「標準のプロパティー」**タブで、BrokerType プロパティーの値フィールドを選択します。ドロップダウン・メニューで、値 ICS、WMQI、または WAS を選択します。
2. 選択したブローカーに関連付けられているプロパティーが「標準のプロパティー」タブに表示されます。ここでファイルを保管するか、または 88 ページの『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。
3. 構成が完了したら、**「ファイル」>「保管」>「プロジェクトに」**を選択するか、または**「ファイル」>「保管」>「ファイルに」**を選択します。

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

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

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

構成ファイル・プロパティーの設定

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

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

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

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

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

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

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

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

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

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

- 背景がグレーのフィールドは、標準のプロパティーを表します。値を変更することはできますが、名前の変更およびプロパティーの除去はできません。
- 背景が白のフィールドは、アプリケーション固有のプロパティーを表します。これらのプロパティーは、アプリケーションまたはコネクターの特定のニーズによって異なります。値の変更も、これらのプロパティーの除去も可能です。
- 「値」フィールドは構成できます。
- **「更新メソッド」**フィールドは通知用であり、構成できません。このフィールドは、値が変更されたプロパティーをアクティブにするために必要なアクションを示します。

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下の手順を実行します。

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

を閉じると、すべてのカテゴリーで入力した値を一括して保管するかまたは破棄するかを確認するプロンプトが出されます。

- 修正した値を保管するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「はい」**をクリックします。「ファイル」メニューまたはツールバーから**「保管」>「ファイルに」**をクリックする方法もあります。

アプリケーション固有の構成プロパティーの設定

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

1. グリッドの左上端の部分で右マウス・ボタンをクリックします。ポップアップ・メニュー・バーが表示されます。プロパティーを追加するときは**「追加」**をクリックします。子プロパティーを追加するには、親の行番号で右マウス・ボタンをクリックし、**「子を追加」**をクリックします。
2. プロパティーまたは子プロパティーの値を入力します。
3. プロパティーを暗号化するには、**「暗号化」**ボックスを選択します。
4. 86 ページの『標準コネクター・プロパティーの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

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

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

コネクター・プロパティーの暗号化

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

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

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

更新メソッド

付録 A『*コネクターの標準構成プロパティー*』の 60 ページの『プロパティー値の設定と更新』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

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

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

ご使用のブローカーが ICS の場合

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

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

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

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

1. ビジネス・オブジェクト・フィールドを選択するため、そのビジネス・オブジェクトの左側の番号をクリックします。
2. 「Connector Configurator」ウィンドウの**「編集」**メニューから、**「行を削除」**をクリックします。リスト表示からビジネス・オブジェクトが除去されます。
3. **「ファイル」**メニューから、**「プロジェクトに保管」**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」ウィンドウでは「エージェント・サポート」の選択の妥当性は検査されません。

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

ほとんどのコネクターの場合、選択可能な項目は「**最大限の努力**」のみです。

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

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

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

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

「**メッセージ・セット ID**」は、WebSphere Business Integration Message Broker 5.0 のオプション・フィールドですが、ここに ID を入力する場合は、一意の ID を入力する必要はありません。ただし、WebSphere MQ Integrator 2.1 や WebSphere MQ Integrator Broker 2.1 の場合は、一意の **ID** を入力する必要があります。

ご使用のブローカーが WAS の場合

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

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

System Manager を実行している場合、「サポートされているビジネス・オブジェクト」タブの「ビジネス・オブジェクト名」列の下にある空のボックスを選択できま

す。コンボ・ボックスが表示され、コネクターが属する統合コンポーネント・ライブラリー・プロジェクトから選択可能なビジネス・オブジェクトのリストが示されます。このリストから必要なビジネス・オブジェクトを選択します。

関連付けられたマップ (ICS のみ)

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

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

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

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

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的**

  場合によっては、関連付けられたマップを明示的にバインドすることが必要になります。

  明示的バインディングが必要なのは、特定のサポートされるビジネス・オブジェクトに複数のマップが存在する場合のみです。ICS は、ブート時、各コネクターでサポートされるそれぞれのビジネス・オブジェクトにマップを自動的にバイン

ドしようとします。複数のマップでその入力データとして同一のビジネス・オブジェクトが使用されている場合、サーバーは、他のマップのスーパーセットである 1 つのマップを見つけて、バインドしようとします。

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

以下の手順を実行して、マップを明示的にバインドします。

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

リソース (ICS)

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

メッセージング (ICS)

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

トレース/ログ・ファイル値の設定

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

ログとトレースの値を変更するには、以下の手順を実行します。

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

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

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

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

データ・ハンドラー

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

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

構成ファイルの保管

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

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

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

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

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

- WAS: 「*アダプター実装ガイド (WebSphere Application Server)*」

構成ファイルの変更

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

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

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

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

構成の完了

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

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

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

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

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

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

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

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

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

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

特記事項

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

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

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

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Burlingame Laboratory Director
IBM Burlingame Laboratory
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

プログラミング・インターフェース情報

プログラミング・インターフェース情報は、プログラムを使用してアプリケーション・ソフトウェアを作成する際に役立ちます。

一般使用プログラミング・インターフェースにより、お客様はこのプログラム・ツール・サービスを含むアプリケーション・ソフトウェアを書くことができます。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

警告: 診断、修正、調整情報は、変更される場合がありますので、プログラミング・インターフェースとしては使用しないでください。

商標

以下は、IBM Corporation の商標です。

IBM
IBM ロゴ
AIX
CrossWorlds
DB2
DB2 Universal Database
Domino
Lotus
Lotus Notes
MQIntegrator
MQSeries
Tivoli
WebSphere

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

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

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

Adapter for EJB 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 Adapter Framework V2.4.0

Printed in Japan

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