WebSphere Business Integration
Server Express and Express Plus

Adapter for eMatrix ユーザーズ・ガイド

Adapter バージョン 2.2.0

WebSphere Business Integration
Server Express and Express Plus

Adapter for eMatrix ユーザーズ・ガイド

Adapter バージョン 2.2.0

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、 99 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、WebSphere Business Integration Server Express Adapter for eMatrix (5724-G87) バージョン 2.2.0 に適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： WebSphere Business Integration
Server Express
and Express Plus
Adapter for eMatrix User Guide
Adapter Version 2.2.0

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2005.9

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

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

目次

本書について v
対象読者 v
関連文書 v
表記上の規則 v

本リリースの新機能 vii
リリース 2.2.0 の新機能 vii

第 1 章 アダプターの概要 1
アダプター・アーキテクチャー 1
アダプターの動作方法 4
サンプル・シナリオ 6

第 2 章 アダプターのインストール 7
アダプター環境 7
インストール作業の概要 8
アダプターおよび関連ファイルのインストール 8
インストール済みファイルの構造 8

第 3 章 アダプターの構成 11
アダプターの構成 11
複数のコネクター・インスタンスの作成 14
コネクターの始動 15
eMatrix アダプターの始動 16
コネクターの停止 17
ログ・ファイルとトレース・ファイルの使用 17

第 4 章 ビジネス・オブジェクトの理解 19
メタデータの定義 19
ビジネス・オブジェクト構造の概要 20
eMatrix のビジネス・オブジェクト 20
ビジネス・オブジェクトの生成 30

第 5 章 ビジネス・オブジェクト定義の生成 31
eMatrix 用の ODA の概要 31
ビジネス・オブジェクト定義の生成 32
ファイルのアップロード 38

第 6 章 トラブルシューティングおよびエラー処理 41
エラー処理 41
トラブルシューティング 42

付録 A. コネクターの標準構成プロパティー 47
新規プロパティー 47
標準のコネクター・プロパティーの概要 47
標準プロパティーの早見表 49
標準のプロパティー 54

付録 B. Connector Configurator Express 71
Connector Configurator Express の概要 71

Connector Configurator Express の始動 72
System Manager からの Configurator の実行 72
コネクター固有のプロパティー・テンプレートの作成 73
新しい構成ファイルの作成 76
既存ファイルの使用 78
構成ファイルの完成 79
構成ファイル・プロパティーの設定 80
構成ファイルの保管 88
構成ファイルの変更 88
構成の完了 89
グローバル化環境における Connector Configurator Express の使用 89

付録 C. サンプル・シナリオ 91
概要 91
前提事項 91
シナリオのインストール 91
シナリオの実行 96

特記事項 99
プログラミング・インターフェース情報 100
商標 101

本書について

IBM(R) WebSphere(R) Business Integration Server Express および IBM(R) WebSphere(R) Business Integration Server Express Plus は、 InterChange Server Express、関連する Toolset Express、Collaboration-Foundation、および一連のソフトウェア統合アダプターの各コンポーネントで構成されています。Toolset Express のツールを使用して、ビジネス・プロセスを作成、変更、および管理することができます。プリパッケージされたアダプターの中から、アプリケーションにまたがるビジネス・プロセスに合わせて選択できます。標準プロセス・テンプレート CollaborationFoundation により、カスタマイズ・プロセスを迅速に作成できます。

本書では、この特定のソフトウェア統合アダプターのインストール、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。

特に明記がない限り、本書内のすべての情報は、IBM WebSphere Business Integration Server Express および IBM WebSphere Business Integration Server Express Plus の両方を対象としています。「WebSphere Business Integration Server Express」という用語と、これを言い換えた用語は、この両方の製品を指します。

対象読者

本書は、お客様のサイトでコネクターを使用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。

関連文書

この製品に付属する完全な資料セットで、すべての WebSphere アダプター・システムに共通する機能とコンポーネントについて説明します。また、特定のコンポーネントに関する参考資料も含まれています。

http://www.ibm.com/websphere/wbiserverexpress/infocenter のサイトから資料をダウンロード、インストール、および表示することができます。

表記上の規則

本書では、以下のような規則を使用しています。

`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。
イタリック、*イタリック*	変数名または相互参照を示します。
青のアウトライン	オンラインで表示したときにのみ見られる青のアウトラインは、相互参照用のハイパーリンクです。アウトラインの内側をクリックすると、参照先オブジェクトにジャンプします。

\	本書では、ディレクトリー・パスに円記号 (¥) を使用します。すべての WebSphere Business Integration Server Express 製品のパス名は、システム上の、製品のインストール先ディレクトリーを基準とした相対パス名です。
ProductDir	WebSphere Business Integration Server Express Adapters 製品のインストール先ディレクトリーを表します。プラットフォームごとのデフォルトの製品ディレクトリーは ¥IBM¥WebSphereServer です。
{ }	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、パイプで区切られた部分は、選択対象のオプションです。1 つのオプションだけ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・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指定できることを意味します。
< >	1 つの名前の個々のエレメントを互いに区別するために、不等号括弧によって個々のエレメントが囲まれます。例えば、`<server_name><connector_name>tmp.log` のように使用します。

本リリースの新機能

リリース 2.2.0 の新機能

Solaris 7.0 はサポートされなくなりました。

コネクター定義ファイルのフォーマットは、.txt から .xsd に変更されました。

第 1 章 アダプターの概要

本章では、IBM(R) WebSphere(R) Business Integration adapter for eMatrix(R) および関連するシステム・アーキテクチャーについて説明します。

MatrixOne eMatrix(TM) は、プロダクト・ライフ・サイクル・マネージメント・ソリューションです。これを使用すると、プロダクト・ライフ・サイクルに関連したすべての文書およびデータ・フローを中央制御し、それらに対するアクセスをモニターすることができます。Adapter for eMatrix は、eMatrix グローバル・コラボレーション・プラットフォームとその他のエンタープライズ・アプリケーション間の、リアルタイムで双方向の統合を可能にします。

アダプターは、アプリケーション自体とは別個にインストールされる、eMatrix コラボレーション・サーバーと通信します。eMatrix コラボレーション・サーバーのダウンロードおよびインストールの詳細については、eMatrix の資料を参照してください。

統合ブローカーとアダプターの関係の詳細については、「*システム管理ガイド*」を参照してください。

本章の内容は、次のとおりです。

- 1 ページの『アダプター・アーキテクチャー』
- 2 ページの『アーキテクチャーの概要』
- 3 ページの『イベント通知』

アダプター・アーキテクチャー

Adapter for eMatrix は、コネクター、Java Program Object (JPO)、およびアダプター・フレームワークを持った Object Discovery agent (ODA) で構成されています。それらが一緒に動作することによって、eMatrix は、統合ブローカーと情報を交換できるようになります。その結果、会計および CRM のパッケージなどの外部アプリケーションとも通信できるようになります。

このアダプターには、以下の 3 つの基本コンポーネントが含まれます。

- **コネクター**

  データの転送において、統合ブローカーとアプリケーションの間の中継を行います。コネクターはメタデータ主導型です。

- **Java Program Object (JPO)**

  複数存在することもあります。eMatrix 内で動作し、eMatrix データベース内のイベントに関する情報を保管します。

- **Object Discovery Agent (ODA)**

  設計時ツールであり、アプリケーションに照会し、eMatrix の既存ビジネス・オブジェクトのタイプおよび関係を確認します。これらが、WebSphere ビジネス・オ

ブジェクト定義の基礎になります。コネクターは、実行時にこれらの定義を使ってアプリケーション・データをビジネス・オブジェクトに変換します。

アダプターは、eMatrix ビジネス・オブジェクトおよび eMatrix 内の接続を操作します。MQL ステートメントを実行することもあります。MQL コマンドを使用することによって、アダプターは、eMatrix によってサポートされるほとんどすべてのアクションを実行できます。

アダプターは、eMatrix アダプター開発キット (ADK) と呼ばれる Java ベースの API および別個の eMatrix コラボレーション・サーバーを使用して、eMatrix アプリケーションと通信します。eMatrix ADK を使用すると、アダプターは、eMatrix コラボレーション・サーバーを介してアプリケーションに要求を渡すことにより、アプリケーション内のオブジェクトに照会したり、それらを操作したりできます。

アーキテクチャーの概要

アダプターは、統合ブローカーおよびアプリケーションとの間で、データ・フローを取り次ぎます。フローは以下のとおりです。

- 要求処理。統合ブローカー (InterChange Server Express) によって開始されます。
- イベント処理。eMatrix アプリケーションによって開始されます。

これらのフローについて、以下で詳しく説明します。

要求処理

要求処理は統合ブローカーによって開始されます。統合ブローカーはビジネス・オブジェクトを Adapter for eMatrix に渡します。アダプターは、Java ベースの eMatrix ADK を使用して eMatrix コラボレーション・サーバーと通信します。3 ページの図 1 は、データ・フローのアーキテクチャーの概要を示しています。

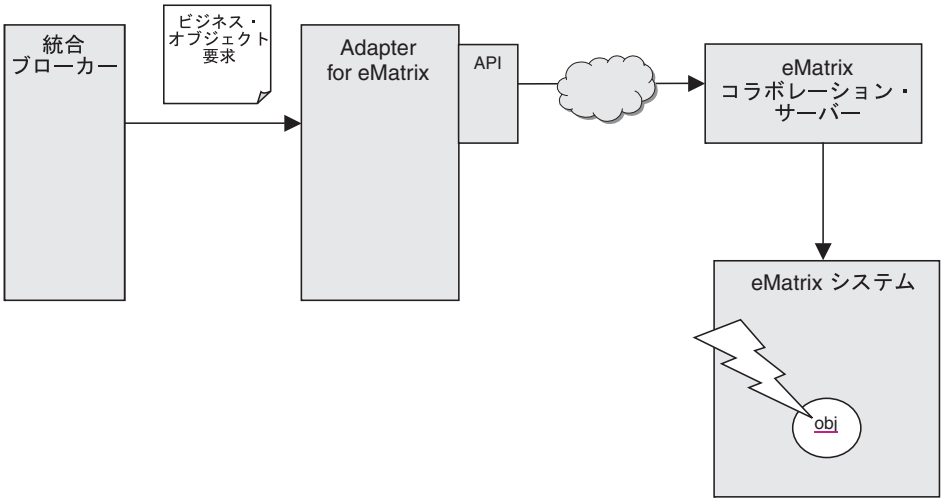

図 1. eMatrix アダプターにおける要求処理

要求処理は以下のように行われます。

1. 統合ブローカーは、ビジネス・オブジェクトの形でアダプターに要求を発行します。このオブジェクトには、タイプ情報およびビジネス・オブジェクトの動詞が含まれています。
2. アダプターは、該当する eMatrix ADK 呼び出しを行うことによって、ビジネス・オブジェクトを処理します。
3. eMatrix ADK 呼び出しが行われるたびに、コラボレーション・サーバーは、指定されたアクションを eMatrix システム内の該当するオブジェクトに対して実行し、結果をコネクターに戻します。
4. 要求の処理を完了すると、アダプターは、状況コードを統合ブローカーに戻します。必要な場合、変更結果を持ったビジネス・オブジェクトも戻します。

イベント通知

Adapter for eMatrix は、実行時にイベント・ポーリングを使用し、eMatrix システム内のイベントを検出します。4 ページの図 2 は、統合ブローカーがアダプターを介してアプリケーションをポーリングしたときに発生する、イベントの順序を示しています。

注: ポーリングを行うには、最初に、モニターしたいイベントの eMatrix トリガーを構成する必要があります。これは、初期セットアップ時に行われます。詳細については、11 ページの『アダプターの構成』を参照してください。

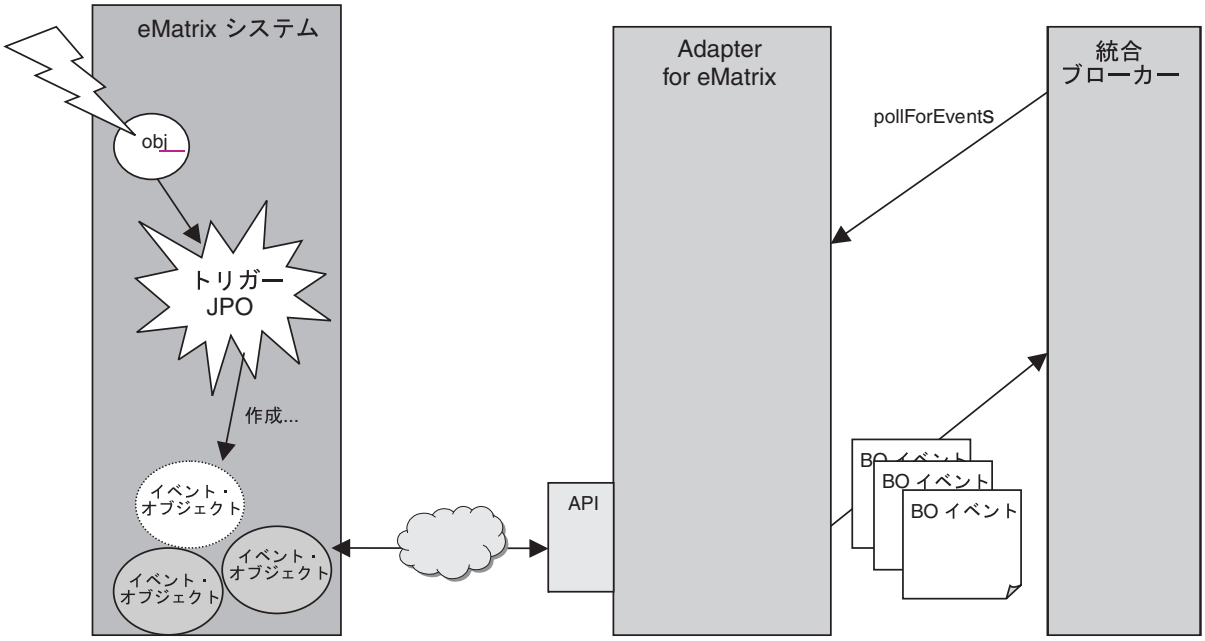

図 2. eMatrix アダプターにおけるイベント通知

イベント通知は以下のように行われます。

1. eMatrix システムでアクションが実行され、トリガーが起動します。
   トリガーはさまざまな方法で起動できます。例えば、構成済みトリガーで eMatrix オブジェクトを変更するなどです。
2. トリガーが起動すると、eMatrix システムは、関連する Java Program Object (JPO) を実行します。これにより、eMatrix 内で新規イベント・オブジェクトが作成されます。
3. アダプターは、指定された間隔で eMatrix をポーリングし、JPO が作成した新規イベント・オブジェクトがないか調べます。
4. アダプターはイベント・オブジェクトを検索します。
5. アダプターは、次に、関連する eMatrix エンティティーを検索し、それらをビジネス・オブジェクトに変換してから、統合ブローカーにパブリッシュします。アダプターは、eMatrix のビジネス・オブジェクトと関係の両方を検索できます。

アダプターの動作方法

Adapter for eMatrix は双方向に動作します。アダプターは、eMatrix システムで発生したイベントを処理し、統合ブローカーによって eMatrix システムに送信された要求を処理することができます。

イベント通知

アダプターは、ポーリング機構によってイベントを検出し、処理します。この機構を使用すると、ユーザーは、イベント優先順位を設定し、イベント検出頻度およびアダプターが処理できるイベント数を制御できます。このレベルの制御によって、大規模システム内でのアダプターの負荷のバランスを取り、重大イベントの経路を

滑らかにできます。これらの値は、アダプターの構成可能プロパティーで設定されます (12 ページの『コネクターの構成』を参照)。

アダプターは、Java Program Object (JPO) および eMatrix ビジネス・オブジェクト定義を使用して、イベントに関する情報の保管および送信を行います。 IBM では、選択して使用できる 2 つのデフォルト JPO を提供しています。この JPO は、以下のとおりです。

- `WBIEventLogger` JPO。イベントを記録するビジネス・オブジェクト・タイプを作成します。
- `InstallEventTables` JPO。イベントを保管するビジネス・オブジェクト・タイプを作成します。

この説明では、デフォルトの WBIEventLogger JPO の使用を想定しています。eMatrix オブジェクトの構造の詳細については、19 ページの『第 4 章 ビジネス・オブジェクトの理解』を参照してください。

イベント通知においては、`WBIEventLogger JPO` が中核をなします。アクションが、特定のビジネス・オブジェクトまたは関係に関連するときは常に、トリガーを定義し、この JPO を呼び出すことができます。JPO は、オブジェクト ID および指定されたオプションを使用して、eMatrix 内で新規イベントのビジネス・オブジェクトまたは関係を作成します。

例えば、「Part Assembly」と呼ばれるオブジェクト・タイプに関連したイベントを追跡したい場合、eMatrix を構成し、そのようなアクションが発生したときに JPO を起動することができます。トリガーが起動すると、JPO は、PartAssembly ID およびイベント中に行われるアクション・タイプを記録します。

トリガー定義の詳細については、11 ページの『JPO の構成』を参照してください。

注: イベント通知は非同期処理です。アダプターは、同期イベント通知をサポートしていません。

イベント検索

アプリケーションをポーリングする場合、アダプターは、未処理イベントを eMatrix に照会します。アダプターは、構成プロパティーの `EventStatus` として READY_FOR_POLL の値を持っているすべてのイベントを検査します。アダプターは、`WBIEventLogger JPO` が作成したイベント・レコードを検索します。

すべてのイベントは同じボールトに存在します。アダプターは、始動時に EventVault プロパティーの値を読み取ります。

アダプターは、次に、関連した eMatrix オブジェクトを検索し、それを統合ブローカーに送信します。イベントが統合ブローカーにパブリッシュされたら、アダプターは、イベントを eMatrix に保存することもできます。

イベントに関する情報は、未処理イベントの場合、「*<prefix>*_Event」、処理済みイベントの場合、「*<prefix>*_Archived_Event」と呼ばれる eMatrix オブジェクトに保持されます。アダプターは、状況コードを使用してイベントの状態を追跡します。

予期せずにアダプターが終了すると、アダプターは、再始動時に「未確定」イベントを報告します。イベントが統合ブローカーに正常にパブリッシュされたかどうかをアダプターが判別できない場合、イベントには未確定というラベルが付けられます。アダプターは、通常のイベントを処理する前に、これらのイベントを処理する必要があります。アダプターが行うアクションの詳細については、41 ページの『エラー処理』を参照してください。

サンプル・シナリオ

Adapter for eMatrix には、アダプターがどのように要求処理およびイベント通知を行うかを示すサンプル・シナリオが含まれています。このシナリオのインストール、構成、および実行については、91 ページの『付録 C. サンプル・シナリオ』を参照してください。

第 2 章 アダプターのインストール

本章では、IBM WebSphere Business Integration Server Express Adapter for eMatrix のインストール方法を説明します。本章の内容は、次のとおりです。

- 『アダプター環境』
- 8 ページの『インストール作業の概要』
- 8 ページの『アダプターおよび関連ファイルのインストール』
- 8 ページの『インストール済みファイルの構造』

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。環境要件は、以下のセクションでリストされています。

- 『アダプターのプラットフォーム』
- 『アダプターの依存関係』
- 8 ページの『グローバリゼーション』

アダプターのプラットフォーム

このアダプターは、次のソフトウェアでサポートされています。

Windows 2000

データベース:

- DB2 7.2
- Oracle 8.1.7.4、Oracle 9.2.0.1

サード・パーティー・ソフトウェア:

- eMatrix Server、バージョン 9.6.01、10.0.0.0
- eMatrix コラボレーション・サーバー、バージョン 9.6.01、10.0.0.0

アダプターの依存関係

アダプター・クライアントのライブラリー/API は、`eMatrixServletRMI.jar` バージョン 10.0.0.0 にあります。

アダプターは、実行時に eMatrix ADK にアクセスする必要があります。 ADK は、eMatrix コラボレーション・サーバーによって分散され、独立型バージョンになっています。

eMatrix コラボレーション・サーバーと同じマシン上にアダプターをインストールしている場合、ADK の独立型バージョンをダウンロードする必要はありません。しかし、アダプターとコラボレーション・サーバーが 2 つの異なるマシン上にある場合、アダプターと同じマシン上に独立型 ADK をインストールする必要があります。

グローバリゼーション

このアダプターは DBCS (2 バイト文字セット) で使用可能であり、翻訳されています。

インストール作業の概要

Adapter for eMatrix をインストールするには、次のステップに従います。

1. 統合ブローカーがインストールされていることを確認します。
2. 必要な前提ソフトウェアがすべてインストールされていることを確認します。
3. リモート eMatrix Server の名前およびロケーションを確認します。
4. Adapter for eMatrix および関連ファイルをインストールします。
5. eMatrix 用アダプターを構成します。

本書のインストールの説明では、ステップ 4 および 5 を対象としています。

アダプターおよび関連ファイルのインストール

WebSphere Business Integration Server Express アダプター製品のインストールについては、「*WebSphere Business Integration Server Express インストール・ガイド (Windows 版)*」を参照してください。このガイドは、WebSphere Business Integration Server Express Adapters Infocenter (http://www.ibm.com/websphere/wbiserverexpress/infocenter) にあります。

インストール済みファイルの構造

表 1 に、アダプターで使用されるファイル構造、およびシステムにインストールされるファイルを示します。

注:

1. 本書では、ディレクトリー・パスとして円記号 (¥) を使用しています。
2. すべての製品パス名は、システム上の、製品のインストール先ディレクトリーを基準とした相対パス名です。

表 1. システムにインストールされたアダプターのファイル

***ProductDir* のサブディレクトリー**	**説明**
¥bin¥Data¥App¥eMatrixConnectorTemplate	eMatrix Connector のテンプレート
connectors¥eMatrix	内容: • CWeMatrix.jar • start_eMatrix.bat
connectors¥eMatrix	以下のコネクター・サービスの始動スクリプトが入っています。 eMatrix_service.bat
connectors¥eMatrix¥dependencies	以下のデフォルトの JPO が入っています。 • InstallEventTables.java • WBIEventLogger.java

表 1. システムにインストールされたアダプターのファイル (続き)

ProductDir のサブディレクトリー	説明
connectors¥eMatrix¥samples	以下のサンプルのコネクター構成ファイルが入っています。 • EmatrixConnector.cfg • PortConnector.cfg • repos • sound_card_update.bo
connectors¥eMatrix¥samples¥repos	以下のサンプルのビジネス・オブジェクトおよび関係オブジェクトが入っています。 • wbi_bundle.xsd • wbi_computer.xsd • wbi_computer_bundle_to_computer.xsd • wbi_computer_computer_to_pci_component.xsd • wbi_computer_to_pci_component.xsd • wbi_cpu.xsd • wbi_cw_event.xsd • wbi_expand_bo.xsd • wbi_input.xsd • wbi_network_card.xsd • wbi_sound_card.xsd • wbi_sound_card_sound_card_to_sound_chip.xsd • wbi_sound_chip.xsd • wbi_usb_2_0_card.xsd
connectors¥messages	エラー・メッセージなどのメッセージがリストされた、eMatrixConnector.txt ファイルが入っています。
ODA¥eMatrix	内容: • eMatrixODA.jar ODA コードを持った Object Discovery Agent • start_eMatrixODA.bat
ODA¥messages	エラー・メッセージなどのメッセージがリストされた、eMatrixODAAgent.txt ファイルが入っています。

第 3 章 アダプターの構成

この章では、Adapter for eMatrix の構成および始動方法を説明します。本章の内容は、次のとおりです。

- 『アダプターの構成』
- 14 ページの『複数のコネクター・インスタンスの作成』
- 15 ページの『コネクターの始動』
- 16 ページの『eMatrix アダプターの始動』
- 17 ページの『コネクターの停止』
- 17 ページの『ログ・ファイルとトレース・ファイルの使用』

アダプターの構成

インストールしたら、始動する前に、以下のようにして、アダプター・コンポーネントを構成します。

JPO の構成

JPO トリガーが動作するには、前もって特定の構成プロパティーが設定されている必要があります。これらの構成プロパティーを設定するには、各 JPO のソース・コードを編集する必要があります。

Adapter for eMatrix と一緒にインストールされる JPO は、`WBIEventLogger` および `InstallEventTables` です。それらをインストールしたら、eMatrix Business Modeler でそれぞれの JPO を探し、ソース・コードを編集する必要があります。

編集する必要があるフィールドは、以下の 11 ページの表 2 にリストされています。

表 2. JPO 構成プロパティー

プロパティー名	説明	デフォルト値
`wbi`*`Prefix`*	eMatrix 内で、すべての WebSphere Business Integration eMatrix メタエンティティー (タイプ、属性、ポリシー) にそれら独自のネーム・スペースを提供するために使用されるプレフィックス。 コネクター構成プロパティーの `wbiPrefix` に指定された値は、`InstallEventTables` ファイルおよび `WBIEventLogger.java` ファイルで宣言された `wbi`*`Prefix`* と一致する必要があります。	`wbi_`
`vault`	新規イベントが作成される eMatrix ボールトの名前。 `EventVault` コネクター構成プロパティーに指定された値に一致する必要があります。	`WBI_Events`
`adapterUser`	eMatrix システムで要求を行うときにアダプターが使用する ID。 `ApplicationUserName` コネクター構成プロパティーに指定された値に一致する必要があります。 「ピンポン」作用を防止するのに重要です (下記参照)。	`adapter`

ピンポン作用の防止

トリガーが割り当てられている eMatrix システム内のオブジェクトをアダプターが変更すると、そのトリガーが新しいイベントを生成し、それがブローカーに戻されてパブリッシュされる場合があります。次の仕組みによって、このことが起こらないようにできます。

JPO は、呼び出されると、特にそれを呼び出したユーザーに関連付けられたユーザー ID を含む、コンテキスト・オブジェクトを受け取ります。このユーザー ID は、JPO プロパティーの adapterUser で設定されます。WBIEventLogger JPO は、起動されると、eMatrix コンテキスト・オブジェクトから受け取ったユーザー ID を検査します。

ユーザーが Adapter for eMatrix の場合、JPO の adapterUser プロパティーの値は、コネクターの ApplicationUserName プロパティーの値に一致します。この場合、JPO はそのイベントを無視し、イベントはログに記録されません。ユーザー ID が一致しない場合、JPO は新しいイベントを作成します。

注: JPO の adapterUser プロパティーとアダプターの ApplicationUserName プロパティーを同じ値に設定することが重要です。そうしない場合は、ピンポン作用が発生します。

コネクターの構成

アダプターのコネクター・コンポーネントには、2 種類の構成プロパティーがあります。1 つは標準の構成プロパティーで、ほとんどのアダプターに適用されます。もう 1 つはアプリケーション固有の構成プロパティーで、ご使用のアダプターにのみ適用されます。コネクターを実行する前に、必ずこれらのプロパティーの値を設定してください。

標準構成プロパティー

標準コネクター構成プロパティーを構成するには、Connector Configurator Express ツールを使用します。詳細については、71 ページの『付録 B. Connector Configurator Express』に記載されています。このツールは、コネクターを構成するためのグラフィカル・ユーザー・インターフェースを提供します。「**標準構成プロパティー (Standard Config Properties)**」タブを選択し、構成プロパティーを追加または変更します。

コネクターの構成プロパティーの値の指定が終了すると、Connector Configurator Express が InterChange Server Express のアダプター・リポジトリーに値を保管します。

コネクターは、始動時に構成値を取得します。実行時のセッション中に、1 つ以上のコネクター・プロパティーの値を変更できます。

- AgentTraceLevel など、一部のコネクター構成プロパティーへの変更は動的であり、即時に有効となります。
- それ以外のコネクター・プロパティーへの変更は静的であり、変更後にコンポーネントまたはシステムの再始動が必要です。

プロパティーが動的と静的のいずれであるかを判別するには、Connector Configurator Express の「更新メソッド」欄を参照してください。

アプリケーション固有の構成プロパティー

アプリケーション固有のコネクター構成プロパティーは、アプリケーションに関連し、実行時にコネクターによって必要とされる情報を提供します。また、コネクター内の静的情報やロジックを、再コーディングや再ビルドを行わずに変更する方法も提供します。

これらのプロパティーを構成するには、Connector Configurator Express 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**」タブを選択します。詳細については、71 ページの『付録 B. Connector Configurator Express』を参照してください。

表 3 には、コネクターのアプリケーション固有の構成プロパティーが、説明および指定可能な値とともにリストされています。

表 3. eMatrix 用のアプリケーション固有の構成プロパティー

プロパティー	説明	指定可能な値	デフォルト値	必須
ApplicationUserName	eMatrix システムに接続するときに使用される名前。	<任意> adapterUser JPO 構成プロパティーに設定された値に一致する必要がある。	WBI	はい
ApplicationPassword	eMatrix システムに接続するときに使用されるパスワード。	<任意>	WBI	はい
EventVault	アダプターがイベントを検索するボールト。	<任意> ボールト JPO 構成プロパティーに設定された値に一致する必要がある。	WBIEventVault	はい
inDoubtEvents	未確定イベントのリカバリー方針。値が指定されないと、アダプターは開始時に失敗する。	以下のいずれか: • FailOnStartup • Reprocess • LogError • Ignore	Reprocess	はい
ArchiveProcessed	True の場合、アダプターはすべてのイベントを保存する。False または未定義の場合、アダプターは、イベントが処理された後に、それらを eMatrix から削除する。	true または false	false	
DefaultVault	アダプターが新規オブジェクトを作成し、既存オブジェクトを操作するデフォルト・ボールト。	<任意>	DefaultWBIVault	
DefaultPolicy	eMatrix に新規オブジェクトを作成するときに、アダプターが使用するデフォルト・ポリシー。	<任意>	DefaultWBIPolicy	
KeepRelations	true の場合、アダプターは、ビジネス・オブジェクトの更新時に関係を維持する。false の場合、アダプターは、更新要求で未定義のすべての関係を破棄する。	true または false	false	

表 3. eMatrix 用のアプリケーション固有の構成プロパティー (続き)

プロパティー	説明	指定可能な値	デフォルト値	必須
UseDefaults	UseDefaults が true に設定される場合、アダプターは、ビジネス・オブジェクト定義でデフォルト設定を使用する。	true または false	true	
PollQuantity	各ポーリング呼び出しで、アダプターが処理できる最大イベント数。	<任意の正の整数>	1	
WBIPrefix	eMatrix システムで、イベント通知に関連付けられたすべてのビジネス・オブジェクトおよび属性の接頭部。 **注:** 既存のビジネス・オブジェクトまたは属性は、この値で開始することはできない。	<任意> JPO wbiPrefix 構成プロパティーに指定された値に一致する必要がある。	wbi_	
EmatrixServer	リモート・ビジネス・オブジェクト・サーバーの名前。	<任意>	:bos	
Hostname	コラボレーション・サーバーの URL。	<任意>	localhost:1099	

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

Windows では、ディレクトリーは ProductDir¥connectors¥connectorInstance です。

コネクターに、コネクター固有のメタオブジェクトがある場合、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

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

InterChange Server Express サーバー名を startup.bat のパラメーターとして指定することができます (例: start_EMATRIX.bat connName serverName)。

ビジネス・オブジェクト定義の作成

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

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

   ```
   ProductDir¥repository¥initialConnectorInstance
   ```

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

コネクター定義の作成

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

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

始動スクリプトの作成

始動スクリプトは以下のように作成します。

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

   dirname

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

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。Windows システムでは、始動スクリプトは、コネクターのランタイム・ディレクトリー *ProductDir*¥connectors¥*connName* (*connName* はコネクターを示します) にあります。

表 4. コネクターの始動スクリプト

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

始動スクリプトは実行時に、デフォルトで *Productdir* (以下のコマンドを参照) 内で構成ファイルを探すようになっています。このディレクトリーに、ご使用の構成ファイルが格納されます。

注: アダプターが JMS トランスポートを使用している場合は、ローカル構成ファイルが必要です。

Windows システムでのコネクターの始動:

- 「**スタート**」メニューから、「**プログラム**」>「**IBM WebSphere Business Integration Express**」>「**アダプター**」>「**コネクター**」を選択します。デフォルトでは、プログラム名は「IBM WebSphere Business Integration Server Express」となっています。ただし、これはカスタマイズすることができます。あるいは、ご使用のコネクターへのデスクトップ・ショートカットを作成することもできます。
- Windows のコマンド行から、start_connName connName brokerName {-cconfigFile} と入力します。
- Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムがブートしたとき (自動サービスの場合)、または Windows サービス・ウィンドウを通じてサービスを始動したとき (手動サービスの場合) に、コネクターが始動します。

eMatrix アダプターの始動

start_eMatrix.bat スクリプトまたは start_eMatrix.sh スクリプトが実行されると、コネクターが始動します。これらの始動ファイルには、アダプターの実行前に設定しておく必要のある 2 つの変数が含まれています。この変数は以下のとおりです。

- EMADK: eMatrix ADK .jar ファイルのロケーションに設定する必要があります。
- EM_LIB: eMatrix 共用ライブラリー・ファイルが格納されたディレクトリーのパスに設定する必要があります。

例えば、次のようになります。

```
set EMADK=c:¥ematrix96¥java¥lib¥eMatrixServletRMI.jar
set EM_LIB=c:¥ematrix96¥bin¥winnt
```

注: 別個のマシン上で eMatrix コラボレーション・サーバーを稼働し、独立型 eMatrix ADK .jar ファイルをそのマシンにインストール済みの場合、EM_LIB を設定する必要はありません。

コネクターの停止

コネクターを停止する方法は、コネクターが始動された方法によって異なります。

Windows:

- そのコネクター用の別個の「コンソール」ウィンドウを作成する始動スクリプトを起動することができます。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。
- Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムのシャットダウン時に、コネクターは停止します。

ログ・ファイルとトレース・ファイルの使用

アダプター・コンポーネントは、いくつかのレベルのメッセージのロギングとトレースを提供します。

コネクターはエラー・メッセージ、通知メッセージ、およびトレース・メッセージを記録するために、アダプター・フレームワークを使用します。エラー・メッセージと通知メッセージは外部ログ・ファイルに記録され、トレース・メッセージとトレース・レベル (0 から 5) はトレース・ファイルに記録されます。デフォルトのメッセージ・ファイル名は `eMatrixConnector.txt` です。

表 5 には、各トレース・レベルで記録されるメッセージのタイプが記載されています。

表 5. トレース・ファイルのトレース・レベル

トレース・レベル	説明
0	• eMatrix ODA からのエラーと致命的エラーを記録します。 • システム管理者の対応が必要な警告を記録します。 • アプリケーションからの重要なメッセージを記録します。
1	メソッドのすべての開始メッセージおよび終了メッセージをトレースします。
2	ODA のプロパティーとそれらの値をトレースします。
3	すべてのビジネス・オブジェクトの名前をトレースします。
4	ビジネス・オブジェクトのプロパティーと、受信した値をトレースします。
5	• すべての ODA プロパティーの ODA 初期化値を示します。 • ビジネス・オブジェクト定義のダンプをトレースします。

ログ・ファイルとトレース・ファイルの名前、およびトレース・レベルを Connector Configurator Express で構成してください。詳細については、71 ページの『付録 B. Connector Configurator Express』を参照してください。

ODA にはロギング機能はなく、エラー・メッセージは直接ユーザー・インターフェースに送られます。トレース・ファイルとトレース・レベルは、Business Object Designer Express で構成されます。このプロセスについては、33 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・レベルはコネクター・トレース・レベルと同一であり、上記の表 5 に記載されています。

エラー処理とトラブルシューティングの詳細については、41 ページの『第 6 章 トラブルシューティングおよびエラー処理』を参照してください。

第 4 章 ビジネス・オブジェクトの理解

この章では、ビジネス・オブジェクトの構造、アダプターがビジネス・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アダプターの前提事項について説明します。

本章の内容は、次のとおりです。

- 19 ページの『メタデータの定義』
- 20 ページの『ビジネス・オブジェクト構造の概要』
- 20 ページの『eMatrix のビジネス・オブジェクト』
- 30 ページの『ビジネス・オブジェクトの生成』

メタデータの定義

eMatrix のアダプターはメタデータ主導型です。WebSphere Business Integration システム内では、メタデータはデータ構造を記述するアプリケーション固有情報であると定義されます。メタデータは、コネクターがビジネス・オブジェクト作成のために実行時に使用する、ビジネス・オブジェクト定義と関係定義を構成するために使用します。

メタデータ主導型アダプターは、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。これによりアダプターは、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できます。ビジネス・オブジェクト定義を変更するには、Business Object Designer Express で編集します。

アプリケーション固有のメタデータは、ビジネス・オブジェクト・タイプ、関係、それらのプロパティー、または属性を表すことができます。各ビジネス・オブジェクトの実際のデータ値は、実行時に送信されます。データ値は、アダプターとブローカーの間で渡されるビジネス・オブジェクトにカプセル化されます。

アダプターには、サポートするビジネス・オブジェクトの構造、親と子のビジネス・オブジェクト間の関係、およびデータのフォーマットに関する前提事項があります。したがって、ビジネス・オブジェクトの構造が eMatrix 内の対応するオブジェクトに定義されている構造と正確に一致することは重要であり、一致しない場合はアダプターがビジネス・オブジェクトを正しく処理できません。

注: ビジネス・オブジェクト構造を変更する必要がある場合、ODA を使用して、eMatrix 内の対応するオブジェクトに変更を加えることをお勧めします。詳細については、31 ページの『第 5 章 ビジネス・オブジェクト定義の生成』を参照してください。

ビジネス・オブジェクト定義の変更の詳細については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

ビジネス・オブジェクト構造の概要

WebSphere Business Integration システムでは、ビジネス・オブジェクト定義の構成は以下のとおりです。

- タイプ名
- サポートされる動詞
- 属性
- アプリケーション固有情報 (ASI)

アプリケーション固有のビジネス・オブジェクト定義は、特殊なタイプのビジネス・オブジェクト定義です。これは、個々のアプリケーションのデータ構造および属性プロパティーを反映します。

一部の属性は、データを格納する代わりに、子ビジネス・オブジェクト、または子ビジネス・オブジェクト配列を指しています。例えば属性は、関係を表す子ビジネス・オブジェクトを参照することができます。

アダプターのビジネス・オブジェクトにはフラットなものと階層構造のものがあります。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、子ビジネス・オブジェクトまたは子ビジネス・オブジェクト配列の両方が格納されます。

階層ビジネス・オブジェクトには、単一カーディナリティーまたは複数カーディナリティーを持つ子ビジネス・オブジェクトを含めることができます。子オブジェクトは、親の属性の中に格納されます。複数カーディナリティーを持つ子ビジネス・オブジェクトを含む属性は、ビジネス・オブジェクトの配列です。単一カーディナリティーを持つ子ビジネス・オブジェクトを表す属性は、このようなオブジェクトを 1 つのみ含むことができます。

eMatrix のビジネス・オブジェクト

WebSphere Business Integration Server Express システム内の有効なアプリケーション固有ビジネス・オブジェクトによって、eMatrix システムの必須のコンポーネントがモデル化されます。以下の eMatrix コンポーネントは、ビジネス・オブジェクトとしてモデル化することができます。

- **ビジネス・オブジェクトのタイプ**:
  情報を編成するために使用されます。ビジネス・オブジェクトのインスタンスが情報を保管します。
- **関係**: 2 つのビジネス・オブジェクト間の関係を定義します。
- **コマンド**:
  カスタム操作の実行または eMatrix プログラムの起動のために使用されます。

ビジネス・オブジェクト定義は、ビジネス・オブジェクト・レベルで定義されたアプリケーション固有情報 (ASI) の `ematrix_class` の値で表されているコンポーネントを指定します。上述の eMatrix コンポーネントに対応する 3 つの有効な `ematrix_class` の値については、次のセクションで説明します。

ビジネス・オブジェクトのタイプ

WebSphere Business Integration Server Express システムでは、eMatrix ビジネス・オブジェクト・タイプは、値が `business_object` の `ematrix_class` を指定するビジネス・オブジェクト・レベルの ASI を含むビジネス・オブジェクト定義によって示されます。また、ASI には、この特定のビジネス・オブジェクトがモデルとなる eMatrix ビジネス・オブジェクト・タイプの名前を指定する `object_type` パラメーターが含まれていなければなりません。例えば、次のようになります。

```
ematrix_class=business_object; object_type=<name of eMatrix business object type>
```

eMatrix ビジネス・オブジェクト・タイプには、2 セットのフィールドが含まれています (従ってシステム内のそれらのビジネス・オブジェクト定義にも含まれます)。フィールド・セットは以下のとおりです。

- プロパティー。固定されており、すべての eMatrix ビジネス・オブジェクト・タイプが共有します。ビジネス・オブジェクト所有者、ポリシー、および改訂などがあります。
- 属性。ユーザーが定義します (WebSphere Business Integration Server Express システム属性と混同しないでください)。

ビジネス・オブジェクトを通じて提供される eMatrix プロパティーが、表 6 にリストされています。

注: String 型の属性を使用して日付を表します。サポートされる日付形式については、eMatrix の資料を参照してください。

表 6. eMatrix ビジネス・オブジェクト・プロパティー

プロパティー	説明	予期される属性タイプ
object_id	eMatrix オブジェクトの一意の ID	String
Name	eMatrix オブジェクトの名前	String
Revision	eMatrix オブジェクト改訂番号	String
Policy	eMatrix オブジェクトの制御ポリシー	String
Owner	eMatrix オブジェクトの所有者	String
State	eMatrix オブジェクトの現在の状態	String
Vault	このオブジェクトを格納する eMatrix ボールト	String
to_relationship	このオブジェクトがサポートする 1 つ以上の「関係先」 eMatrix 関係	eMatrix 関係をモデル化する子オブジェクト Cardinality = 1...n
from_relationship	このオブジェクトがサポートする 1 つ以上の「関係元」 eMatrix 関係	eMatrix 関係をモデル化する子オブジェクト Cardinality = 1...n

eMatrix では、オブジェクト ID を通じて、またはタイプ、名前、および改訂の組み合わせを通じて、ユーザーがオブジェクトを識別します。そのため、すべてのビジネス・オブジェクトは、以下のいずれかを含んでいなければなりません。

- プロパティー `object_id` にマップされる属性
- プロパティー `name` および `revision` のそれぞれにマップされる 2 つの属性

これらは、ビジネス・オブジェクト定義内のキー値となります。例については、22 ページの『ビジネス・オブジェクト定義の例』を参照してください。

特定のビジネス・オブジェクト属性が eMatrix 属性にマップされるように指定するためには、以下を実行します。

1. 属性を、string または integer などの単純タイプとして定義します。
2. ビジネス・オブジェクト定義属性レベル ASI に、

   `attr=<eMatrix attribute name>`

   と記述します。

ビジネス・オブジェクト定義の例

22 ページの表 7 に、`PartNumber` および `Size` の 2 つのユーザー定義 eMatrix 属性を含み、ASI プロパティー `object_id` にマップされるビジネス・オブジェクト定義を示します。

表 7. ビジネス・オブジェクト定義 #1

属性名	タイプ	デフォルト値	アプリケーション固有情報	IsKey
ObjectId	String		prop=object_id	はい
State	String		prop=state	いいえ
PartNumber	String	ASX31	attr=PartNumber	いいえ
Size	Integer	15	attr=Size	いいえ

22 ページの表 8 に示すのは同じビジネス・オブジェクト定義ですが、eMatrix プロパティーの `name` および `revision` にマップされています。

表 8. ビジネス・オブジェクト定義 #2

属性名	タイプ	デフォルト値	アプリケーション固有情報	IsKey
Name	String	AllSeasonX31	prop=name	あり
Revision	String	1	prop=revision	あり
State	String		prop=state	なし
PartNumber	String	ASX31	attr=PartNumber	なし
Size	Integer	15	attr=Size	なし

関係

WebSphere Business Integration Server Express システムでは、eMatrix 関係は、値が `relationship` の `ematrix_class` を指定するビジネス・オブジェクト・レベルの ASI を含むビジネス・オブジェクト定義によって示されます。また、ASI には、この特定のビジネス・オブジェクトがモデルとなる eMatrix 関係の名前を指定する `object_type` パラメーターが含まれていなければなりません。例えば、次のようになります。

`ematrix_class=relationship; object_type=<name of eMatrix relationship type>`

前述のビジネス・オブジェクト定義と同様に、eMatrix 関係を反映するビジネス・オブジェクト定義には、eMatrix プロパティー (固定) および属性 (ユーザー定義) の組み合わせを含めることができます。

23 ページの表 9 に、関係プロパティーをリストします。

表 9. eMatrix 関係オブジェクト・プロパティー

プロパティー	説明	予期される属性タイプ
relationship_id	eMatrix 関係の一意の ID	String
to_object	この関係の関係先の eMatrix オブジェクト	eMatrix ビジネス・オブジェクトをモデル化する子オブジェクト Cardinality = 1
from_object	この関係の関係元の eMatrix オブジェクト	eMatrix ビジネス・オブジェクトをモデル化する子オブジェクト Cardinality = 1

すべての関係ビジネス・オブジェクト定義には、以下のデータが含まれていなければなりません。

- プロパティー `relationship_id` にマップする属性
- `to_object` または `from_object` 属性のインスタンス (下記の注を参照)
- ビジネス・オブジェクトを `to_object` または `from_object` 子ビジネス・オブジェクト・コンテナーとして分類するアプリケーション固有情報を持つ属性

注: 単一の関係が多くのオブジェクト・タイプをサポートできるため、関係オブジェクトには、`to_object` または `from_object` 属性のインスタンスが多く含まれています。しかし、関係が、親ビジネス・オブジェクトの `to_relationship` として定義される場合、関係オブジェクトは `to_object` の ID のみを持つことができます。同様に、親ビジネス・オブジェクトで `from_relationship` が定義される場合には、関係オブジェクトは `from_object` の ID のみを持つことができます。

関係定義は、以下のいずれかの書式で記述します。

```
--Business_Object_A
      | -- to_relationship = Relationship_A_to_B
                      | -- to_object = Business_Object_B
```

または

```
--Business_Object_A
      | -- from_relationship = Relationship_A_to_B
                      | -- from_object = Business_Object_B
```

例については、24 ページの『関係の例』を参照してください。

関係の定義

アダプターは、ビジネス・オブジェクト定義で明示的にその関係を定義した場合のみ、異なるオブジェクト間の関係を認識できます。ビジネス・オブジェクト同士が関連する場合、アダプターは、ビジネス・オブジェクトを接続するために関係オブジェクトを必要とします。ビジネス・オブジェクト同士が関係オブジェクトによって接続されていない場合、関連しないものと見なされます。

例えば、次のようなビジネス・オブジェクトを定義します。

```
--Business_Object_A
      | -- to_relationship = Relationship_A_to_B
                     | -- to_object = Business_Object_B
```

そして親ビジネス・オブジェクト (オブジェクト A) で Retrieve を実行した場合、アダプターは A を取り込み、また、A から B への関係を通じて A に関係付けられているタイプ B のオブジェクト**のみ**を取り込みます。

ただし、次のビジネス・オブジェクトでは異なる結果になります。

```
--Business_Object_D
      | -- Business_Object_E
```

このとき、親ビジネス・オブジェクト (オブジェクト D) で Retrieve を実行すると、アダプターは D を取り込み、また、D に関連するかどうかにかかわらずタイプ E のオブジェクトを取り込みます。この場合、ビジネス・オブジェクト E は、有効な検索基準を含む属性を持っていなければなりません。持たない場合はエラーとなります。

ビジネス・シナリオおよび関係に応じて、関連するビジネス・オブジェクトと関連のないビジネス・オブジェクトを定義中で結合することができます。例えば、以下のようなビジネス・オブジェクトを定義することができます。

```
--Business_Object_A
      | -- to_relationship = Relationship_A_to_B
                     | -- to_object = Business_Object_B
                   | -- Business_Object_D
```

ここで、ビジネス・オブジェクト A および B は互いに関連していますが、ビジネス・オブジェクト D はそのどちらにも関連していません。

関係の例

以下の 2 つの例では、それぞれのビジネス・オブジェクト定義内で関係の定義は異なっていますが、同じ方法で関連付けられています。

```
--Car
      | -- to_relationship = CarToTireRelationship
                     | -- to_object = Tire

--Tire
      | -- from_relationship = CarToTireRelationship
            | -- from_object = Car
```

コマンド・オブジェクト

WebSphere Business Integration Server Express システムでは、eMatrix コマンドは、`command` という値を使用して `ematrix_class` を指定する、ビジネス・オブジェクト・レベルの ASI を含むビジネス・オブジェクト定義で表現されます。例えば、次のようになります。

```
ematrix_class=command
```

コマンド・ビジネス・オブジェクト定義は、以下のような構造になっています。

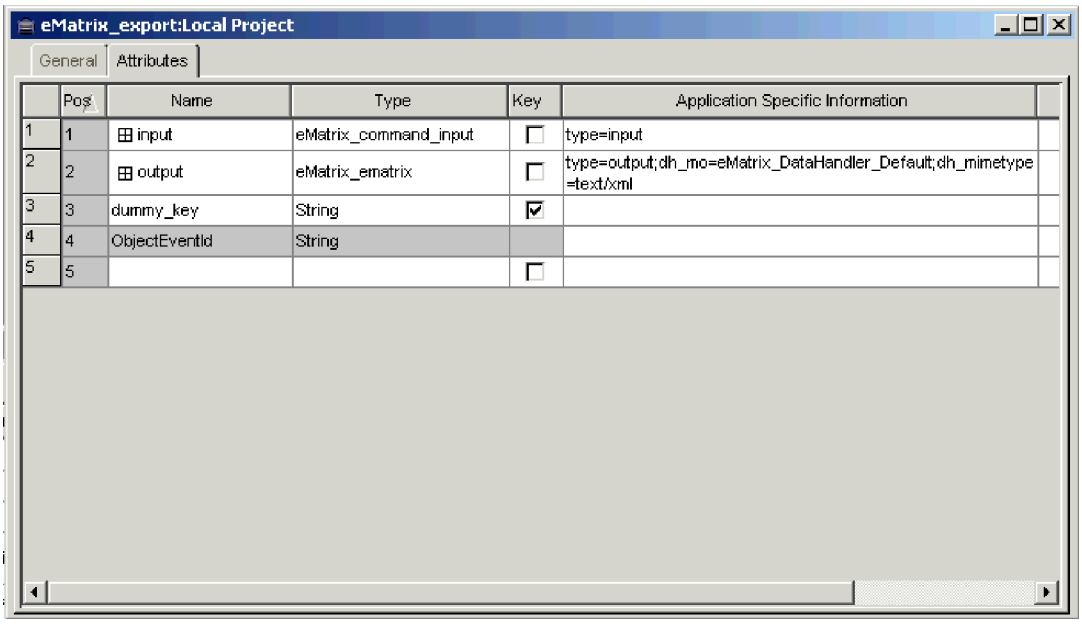

図 3. コマンド・ビジネス・オブジェクト定義の構造

コマンド・オブジェクトには 2 つの属性があります。1 つはコマンドの入力を表し、もう 1 つはコマンドの出力を表します。入力属性は必須であり、オブジェクトである必要があります。出力属性はオプションであり、ストリングまたはオブジェクトです。入力属性および出力属性は、対応する属性レベル ASI で識別されます (`type=input` または `type=output`)。

入力オブジェクトには、実行されるコマンド・ストリングと複数の変数が含まれ、この変数の値がコマンドに挿入されます。コマンド・ストリングは入力オブジェクトのビジネス・オブジェクト・レベル ASI に入れられ、`command=` プレフィックスで識別されます。次のスクリーン・ショットは、コマンドと入力変数を含む入力オブジェクトを示します。

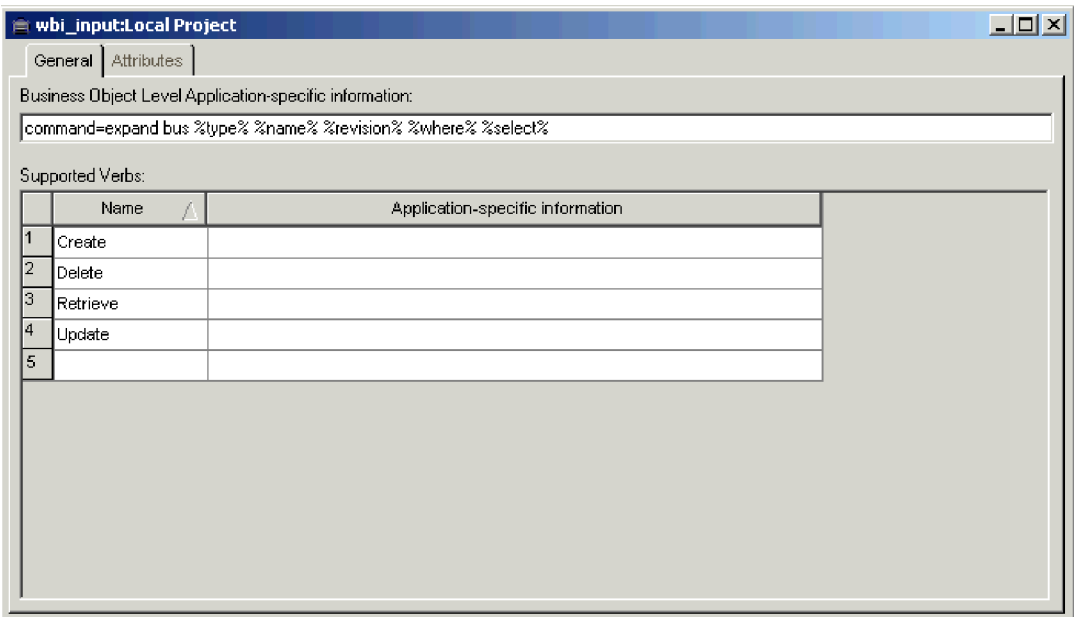

図 4. 入力オブジェクト・コマンド

このスクリーン・ショットには、expand bus コマンドが含まれています。入力オブジェクトの属性は、MQL インタープリターに送信される前にこのコマンド・ストリングに挿入される変数を表します (図 5 を参照)。

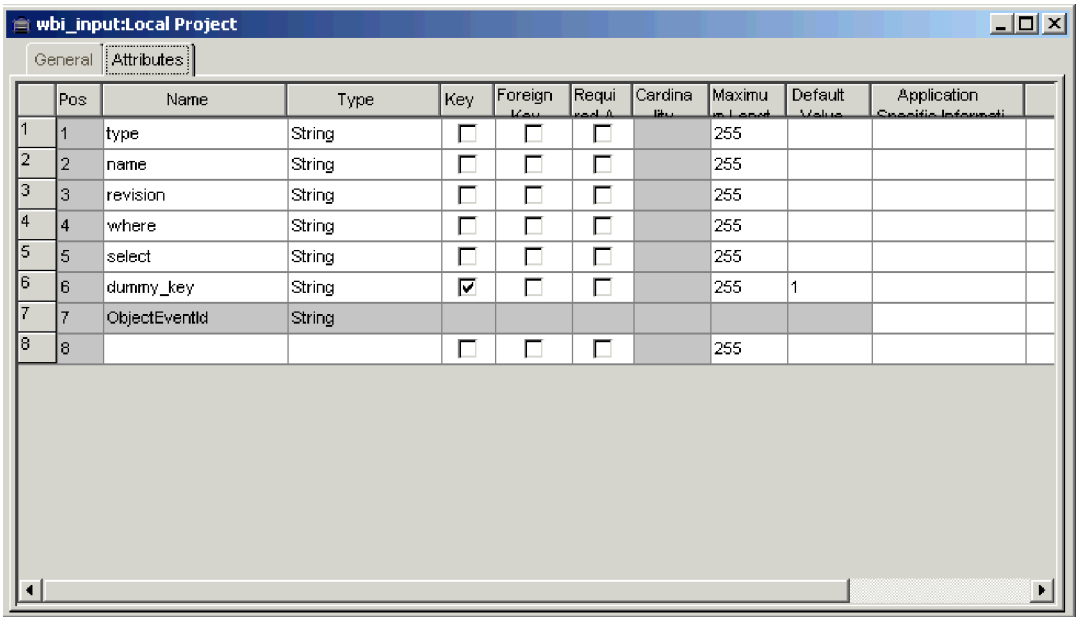

図 5. 入力オブジェクト属性

出力属性はストリングまたはビジネス・オブジェクトです。ビジネス・オブジェクトの出力属性がストリングの場合、出力属性にはコマンドによって戻されるテキストが含まれます。出力属性がオブジェクトの場合、コマンドの結果はデータ・ハンドラーを介して渡され、その後ビジネス・オブジェクト階層に変換されます。この場合、コマンド・オブジェクトの出力属性にはデータ・ハンドラーを起動する情報

を含める必要があります。出力属性の ASI には、次のフィールドが追加される場合があります。

表 10. ASI 出力属性

ASI 出力属性	説明
dh_mo	データ・ハンドラー・メタオブジェクト名を指定します
dh_mimetype	データ・ハンドラーに渡される MIME タイプ・コンテンツを指定します
dh_classname	完全修飾データ・ハンドラー Java クラス名を指定します

次のスクリーン・ショットは、出力属性と、その対応する出力オブジェクトの例を示します。これらのスクリーン・ショットは、前出の入力オブジェクトのスクリーン・ショットとは関連していません。

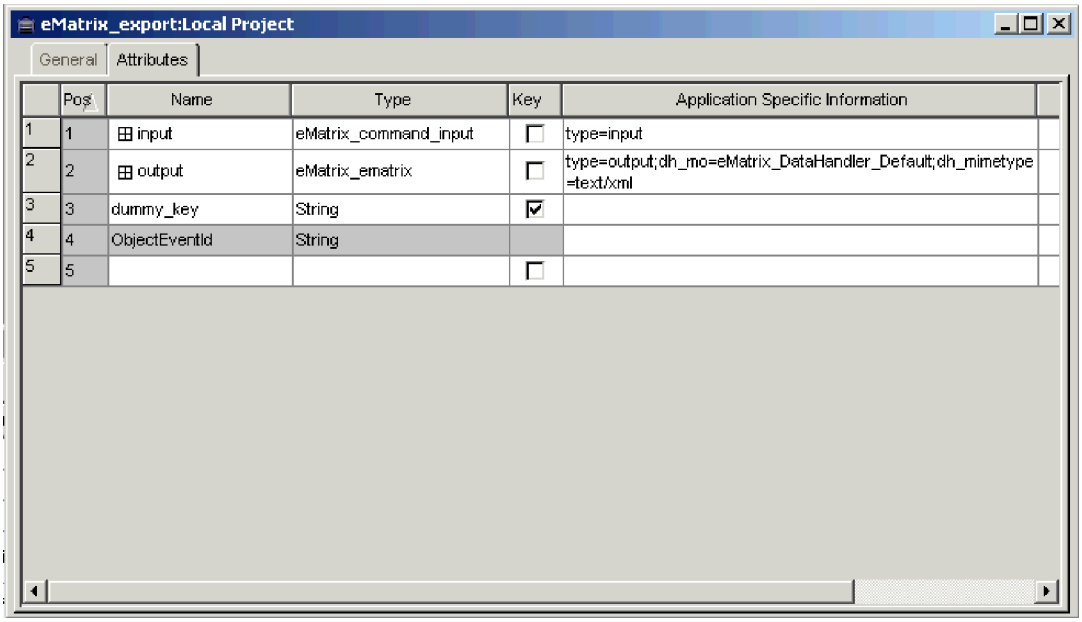

図 6. コマンド・オブジェクト

注: 図 6 では、出力属性は属性レベル ASI `type=output` で識別されます。また、ASI には XML データ・ハンドラーを識別して起動する追加情報が含まれています。

注: 上記で示される入力オブジェクトおよび出力オブジェクトは、コマンド・オブジェクトの子オブジェクトではありません。

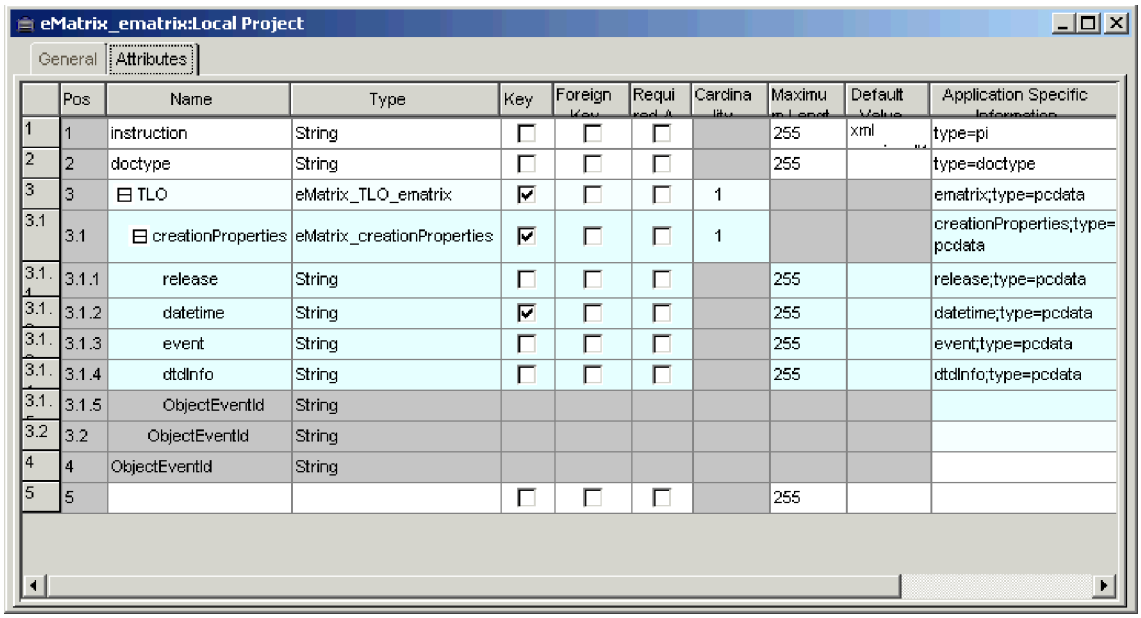

図 7. コマンド出力をキャプチャーするように割り当てられたビジネス・オブジェクト階層

注: 上の図で示されるビジネス階層は、MQL コマンドの出力をカプセル化する、単純 XML DTD を表します。

注: コマンド・ビジネス・オブジェクトでサポートされる動詞は Create のみです。

ビジネス・オブジェクトの動詞

すべてのビジネス・オブジェクトは動詞を含んでおり、この動詞は、アプリケーション固有のビジネス・オブジェクト内のデータを eMatrix アダプターが処理する方法を記述しています。

アダプターは、一意のオブジェクト ID を用いるか、またはタイプ、名前、および改訂の組み合わせを用いて、ビジネス・オブジェクトを識別します。これらはすべてユーザー定義のプロパティーです。詳細については、21 ページの『ビジネス・オブジェクトのタイプ』を参照してください。

アダプターは、eMatrix オブジェクトのクラスが異なる場合、同じ動詞について異なる解釈をします。アダプターは、ビジネス・オブジェクト定義内の `ematrix_class` の値をチェックしてから、ビジネス・オブジェクトを処理します。

28 ページの表 11 に、Adapter for eMatrix で使用可能なビジネス・オブジェクト動詞をリストし、それぞれの動詞でアダプターが実行するアクションを説明します。

表 11. ビジネス・オブジェクト動詞とアクション

動詞	eMatrix クラス	アクション
Create	ビジネス・オブジェクト	eMatrix 内で、1 つ以上の新規ビジネス・オブジェクトを関係と共に作成します。要求で指定された名前、タイプ、および改訂を使用します。ボールトとポリシーが指定されていればそれらを使用し、指定されていなければデフォルトを使用します。 指定された子オブジェクトと関係を作成します。

表 11. ビジネス・オブジェクト動詞とアクション (続き)

動詞	eMatrix クラス	アクション
	関係	すべての子オブジェクトは親オブジェクトと同じ動詞を持つ必要があるため、関係をそれぞれ独立して作成することはできません。すでに関係が存在しているオブジェクト間に新規の関係を作成するには、Update を使用します。
	コマンド	eMatrix システムでコマンドを実行し、要求オブジェクトに結果をすべて戻します。
Update	ビジネス・オブジェクトおよび関係	1 つ以上の指定されたビジネス・オブジェクトおよびそれらの関係を更新します。例については後述します。
Delete	ビジネス・オブジェクト	指定されたビジネス・オブジェクトを削除し、それに結び付けられた関係を自動的に削除します。すべての削除は物理的削除です。
	関係	関係 ID が示す関係を削除します。その関係に結び付くビジネス・オブジェクトの接続を解除しますが、ビジネス・オブジェクトは削除しません。
Retrieve	ビジネス・オブジェクトおよび関係	関係と、ビジネス・オブジェクト内の属性に対応する eMatrix ビジネス・オブジェクトを含む、ビジネス・オブジェクト全体を、eMatrix から取り込みます。統合ブローカーに戻されるビジネス・オブジェクトは、ビジネス・オブジェクト定義内の構造を正確に反映していなければなりません。
RetrieveBy Content	ビジネス・オブジェクトおよび関係	キー情報または非キー情報、あるいはその両方を使用して、eMatrix からビジネス・オブジェクト全体を取り込みます。一致が複数検出された場合は、最初の一致を戻し、「複数のヒット」を知らせるメッセージを送信します。
Exists	ビジネス・オブジェクトおよび関係	トップレベル・ビジネス・オブジェクトが eMatrix 内に存在するかどうかを確認します。

動詞は、アプリケーション固有のビジネス・オブジェクト内部に含まれています。ビジネス・オブジェクトを作成するブローカーによって値が設定されます。アダプターは、動詞およびビジネス・オブジェクトの内容に応じて、eMatrix アプリケーション上で異なる操作を実行します。

アダプターは、ビジネス・オブジェクトを処理し、eMatrix システム内で必要なアクションを起動すると、状況コードを戻します。また、アプリケーションの状態に関する情報を含む新規ビジネス・オブジェクトを統合ブローカーに戻すこともあります。

属性プロパティー

ビジネス・オブジェクト定義には、それらのビジネス・オブジェクト属性で設定可能な、さまざまなプロパティーがあります。以下の表に属性とその値を示します。

表 12 には単純属性をリストしています。

表 12. 単純属性

属性	プロパティー
Name	ビジネス・オブジェクトのフィールド名を指定します。
Type	ビジネス・オブジェクトのフィールド・タイプを指定します。
MaxLength	デフォルトは 255 です。

表 12. 単純属性 (続き)

属性	プロパティー
IsKey	各ビジネス・オブジェクトは、少なくとも 1 つのキー属性を持つ必要があります。キー属性は、属性のキー・プロパティーを true に設定することで指定します。
IsForeighKey	使用されません。
Is Required	使用されません。
AppSpecInfo	attr=<attribute name> などの、アプリケーション固有情報 (ASI) を格納します。
DefaultValue	必要な属性が設定されていない場合に、コネクターがインバウンド・ビジネス・オブジェクト内の単純属性用に使用するデフォルト値を指定します。

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 (ASI) 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指示を含むコネクターを提供します。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、ビジネス・オブジェクトおよび各ビジネス・オブジェクト属性で指定することができ、各動詞ごとに指定することもできます。単純属性の場合、ASI は eMatrix 属性のタイプまたは eMatrix プロパティー (名前、改訂、ボールトなど) の名前のいずれかを指定します。子ビジネス・オブジェクトの場合、ASI は (eMatrix ビジネス・オブジェクトおよび関係を表すオブジェクト階層の) 関係の方向を指定します。

ビジネス・オブジェクトの生成

Business Object Designer Express の提供するグラフィカル・インターフェースを使用すると、実行時に使用するビジネス・オブジェクト定義を生成できます。詳細については、31 ページの『第 5 章 ビジネス・オブジェクト定義の生成』を参照してください。

eMatrix 内のオブジェクト・モデルが変更された場合は、Business Object Designer Express の ODA を使用して、更新を反映するようにビジネス・オブジェクト定義を変更するか、または新規の定義を作成します。

第 5 章 ビジネス・オブジェクト定義の生成

この章では、Object Discovery Agent (ODA) for eMatrix の詳細と、それを使用して Adapter for eMatrix のビジネス・オブジェクト定義を生成する方法について説明します。

本章の内容は、次のとおりです。

- 31 ページの『eMatrix 用の ODA の概要』
- 32 ページの『ビジネス・オブジェクト定義の生成』
- 38 ページの『ファイルのアップロード』

eMatrix 用の ODA の概要

ODA (Object Discovery Agent) を使用すると、アプリケーション固有のビジネス・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 Express 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。

Object Discovery Agent (ODA) for eMatrix は、ビジネス・オブジェクトおよび関係タイプの両方をサポートします。ODA は、Business Object Designer Express で指定する初期設定プロパティーを使用して、eMatrix データベースに接続します。

ODA は次に、eMatrix データベースのすべてのビジネス・オブジェクト・タイプ名を検索し、Business Object Designer Express にそれらを表示します。任意のビジネス・オブジェクト・タイプを展開し、それらに関連した関係を見ることができます。また、関係を展開することもできます。次に、1 つ以上のビジネス・オブジェクト・タイプ (親または子) および関係タイプを選択し、そのビジネス・オブジェクト定義を生成することができます。

Business Object Designer Express ウィザードは、ODA によって検索された eMatrix ビジネス・オブジェクト属性を使用して、定義作成プロセスを自動化します。子ビジネス・オブジェクトおよび関係タイプ用に作成されたビジネス・オブジェクト定義は、親ビジネス・オブジェクト定義に関連付けられます。ビジネス・オブジェクト定義はサーバーに保管する前に表示または修正できます。

ビジネス・オブジェクト定義を生成するために、ODA を次の 2 つの段階で使用できます。

1. **システムが最初にセットアップされるとき:** eMatrix 内で定義されるすべてのイベントについて、一組のビジネス・オブジェクト定義を作成する必要があります。
2. **イベントで使用されるオブジェクトまたは関係タイプが eMatrix で変更されるとき (常に):** Business Object Designer Express で ODA を実行し、変更されたオ

ブジェクトのメタデータを選び出し、新しいビジネス・オブジェクト定義をブローカー・リポジトリーにエクスポートします。

ビジネス・オブジェクト定義の生成

このセクションでは、Business Object Designer Express で eMatrix 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 Express の起動方法と使用方法の詳細については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

始める前に

eMatrix ODA を実行する前に、eMatrix RMI コラボレーション・サーバーをインストールする必要があります。コラボレーション・サーバーがインストールされると、ODA 用の RMI デーモンを実行して、サーバーに接続する必要があります。

ODA とサーバーは、異なるマシン上で実行できます。その場合、リモート・マシン上のコラボレーション・サーバーのホスト名、ポート、および名前を知っている必要があります。

ODA およびコラボレーション・サーバーが別個のマシン上で稼働しているかどうかには関係なく、RMI サーバーと ODA 接続ストリングが同じポート上で稼働していることを検証します。これを行うには、以下のいずれかを実施します。

- サーバーのインストール・ディレクトリー `eMatrix¥bin¥winnt` に移動し、`rmireg.bat` ファイルをダブルクリックします。これによって、RMI サーバーが始動され、ポップアップ・ウィンドウにポート番号が表示されます。
- *`RMIServerInstallationDir`*`¥java¥properties` ディレクトリーにある `framework.properties` ファイルを開き、`ematirx.server.host` プロパティーを調べます。このプロパティーには、ホスト名とポートの両方がリストされます。

ODA の始動

eMatrix 用の ODA のデフォルト名は `eMatrixODA` です。この名前は、開始スクリプト内の `AGENTNAME` 変数の値を変えることで変更できます。

- Windows から ODA を始動するには、次のように選択します。

  「スタート」>「プログラム」>「IBM WebSphere Business Integration Adapters」
  「アダプター」>「Object Discovery Agent」>「eMatrix Object Discovery Agent」
- このコマンド行から ODA を始動するには、次のコマンドを実行します。

  ```
  start_eMatrixODA
  ```

Business Object Designer Express の実行

Business Object Designer Express 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。手順は以下のとおりです。

Business Object Designer Express の始動

Windows から Business Object Designer Express を実行するには、以下のようにします。

1. **「スタート」>「プログラム」>「IBM WebSphere Business Integration Express Adapters」>「ツール」>「Business Object Designer Express」を選択します。**
2. 「Business Object Designer Express」メイン・ウィンドウが開きます。
3. ODA を選択するには、**「ファイル」>「ODA を使用して新規作成」**を選択します。*「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 (33 ページの図 8 を参照)。

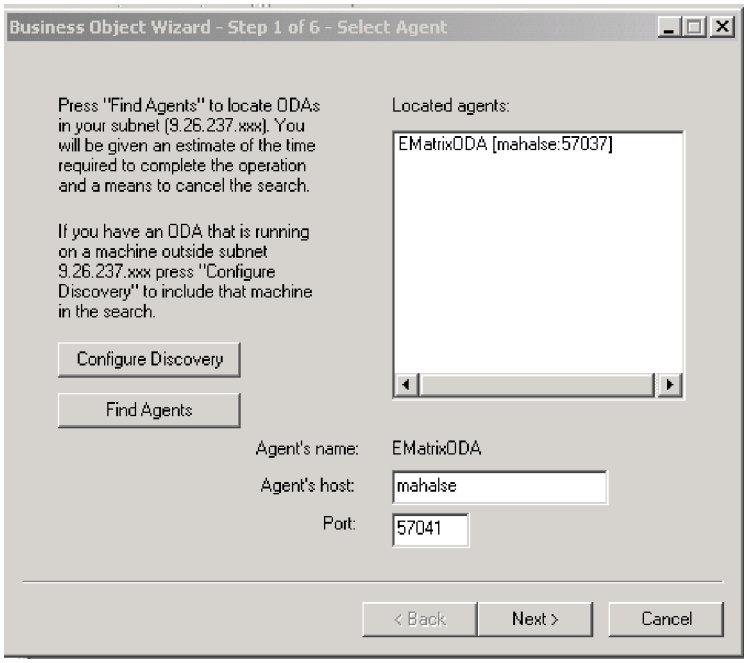

図 8. 「エージェントの選択」画面

4. **「エージェントの検索」**をクリックします。eMatrixODA を含め、システムで実行されている ODA が検索され、**「検索されたエージェント:」**ウィンドウに表示されます。
5. eMatrixODA を選択して、**「次へ」**を選択します。

エージェントの構成

*「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。

この画面で、eMatrix データベースと通信するコンテキストを設定するのに ODA が必要とする情報を入力します。必要なプロパティーおよびそれらの値が 34 ページの表 13 に示されています。

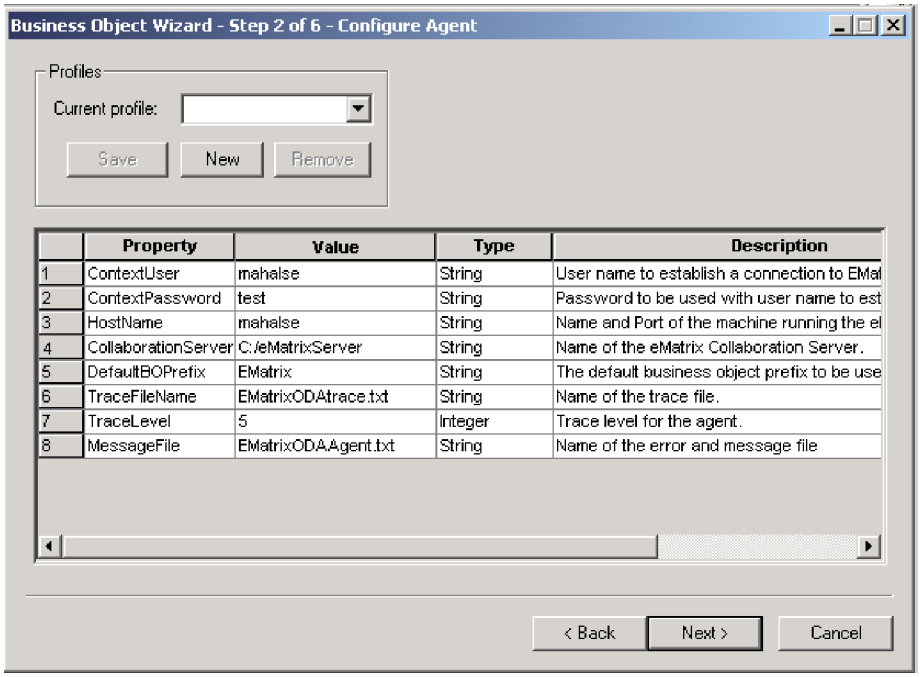

図 9. 「エージェントの構成」画面

1. 「プロファイル」メニューを使用して新規プロファイルを作成します。既存プロファイルを選択することもできます。
2. 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と、プロパティー値が入力されます。

表 13. ODA 構成用のプロパティー値

プロパティー名	プロパティー・タイプ	デフォルト値	説明	必須
ContextUser	String	Creator	eMatrix データベースへ接続するコンテキスト・オブジェクトの設定に使用されるユーザー名。	はい
ContextPassword	String	なし	eMatrix データベースへ接続するコンテキスト・オブジェクトの設定に使用されるパスワード。空にしておくこともできます。	はい
HostName	String		eMatrix コラボレーション・サーバーのホスト名およびポート。`katari:1099` など。	はい
DefaultBOPrefix	String	EMatrix	ビジネス・オブジェクト定義のデフォルトの接頭部。	いいえ
TraceFileName	String	<agentname> Trace.txt	トレース・ファイルの名前。	いいえ
TraceLevel	Integer	5	ODA に適用されるトレース・レベル。	はい
MessageFile	String	<agentname> Agent.txt	エラー・ファイルおよびメッセージ・ファイルの名前。	はい

eMatrix ODA によって表示されるすべてのメッセージは、標準メッセージ・ファイル・フォーマットの `eMatrixODAAgent.txt` ファイルに表示されます。

注: メッセージ・ファイルの名前が正しく指定されていない場合には、ODA はメッセージなしで実行されます。

プロファイルの作成

この画面に入力した値すべてをプロファイルに保管することもできます。ODA を次回に実行する際に、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ておいた値を使用すれば、すべてのデータを再入力する必要はありません。

同じプロパティーに異なる値が指定された、複数のプロファイルを保管することができます。

入力が完了したら、「**次へ**」をクリックします。

ビジネス・オブジェクトの選択

「*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。この画面を使用して、ODA がビジネス・オブジェクト定義を生成する、任意の数のビジネス・オブジェクトまたは関係タイプを選択します。

画面には、eMatrix で定義されているタイプがリストされています。eMatrix が初期化されていない場合には、リストは空になります。

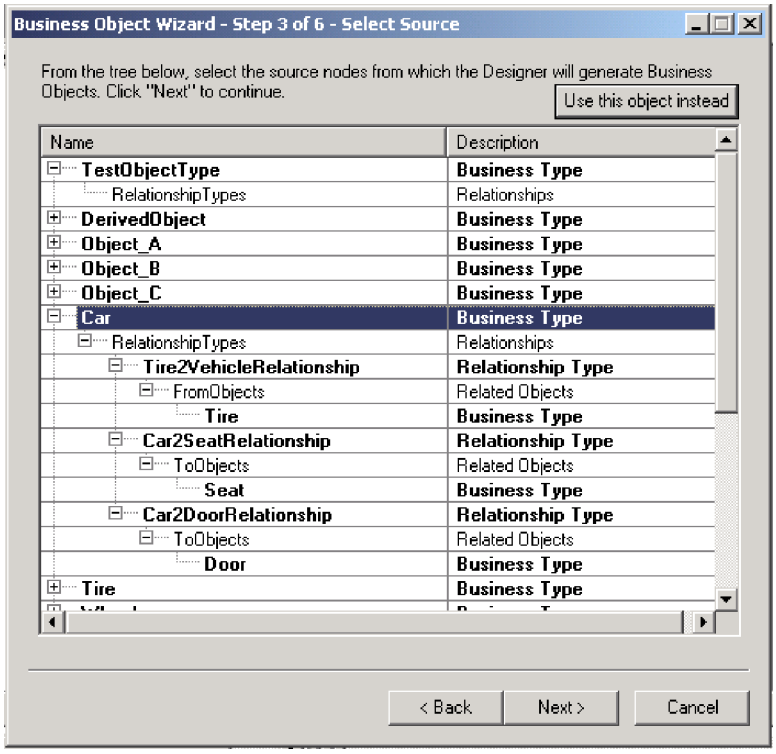

図 10. 「ソースの選択」画面

- 親ビジネス・オブジェクトを展開して、その関係タイプのリストを見ます。
- 関係タイプを展開して、関連しているビジネス・オブジェクトのリストを見ます。

1. 使用するビジネス・オブジェクトのタイプを選択します。

2. 「**次へ**」をクリックします。

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。ここには、選択したソース・ノードが表示されます。

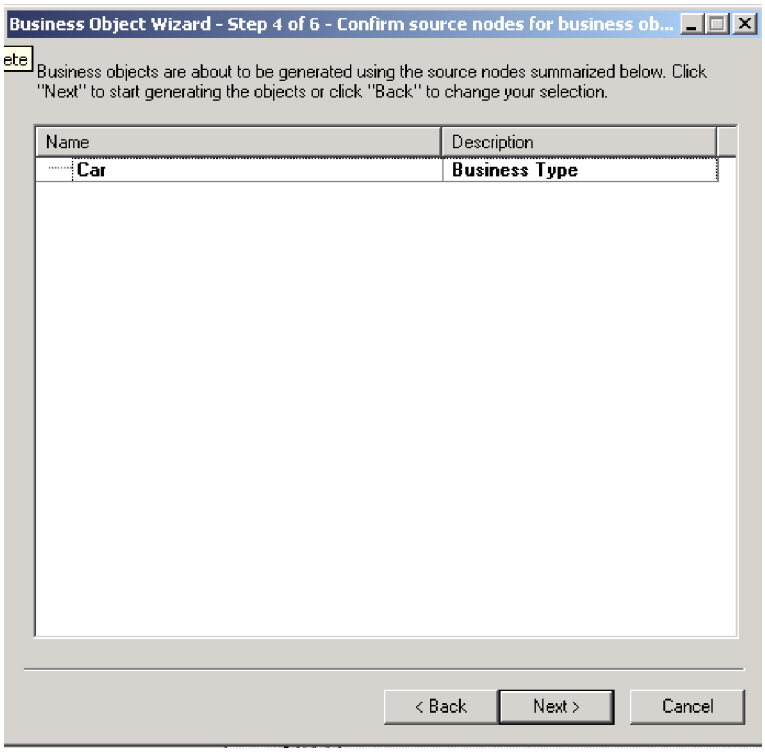

図 11. ソース・ノード画面を確認します。

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

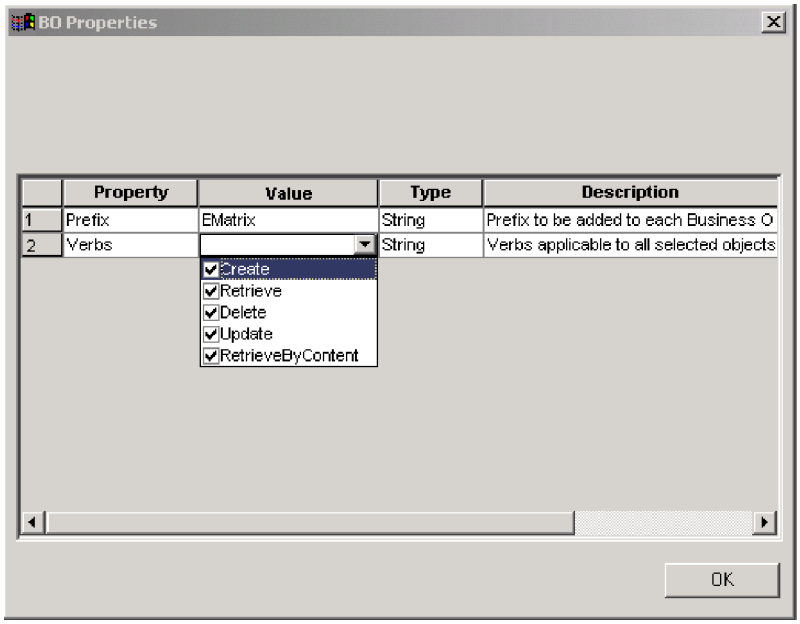

図 12. 「ビジネス・オブジェクトの生成中」画面

次の画面で、ビジネス・オブジェクト・プロパティーを設定する必要があります。ここで、ビジネス・オブジェクトが使用する、ビジネス・オブジェクトの接頭部および動詞を選択できます。複数の動詞を選択することもできます。ビジネス・オブジェクトに使用させたくない動詞の値を選択解除できます。

「**OK**」をクリックし、続けます。

ビジネス・オブジェクト定義の保管

「*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。

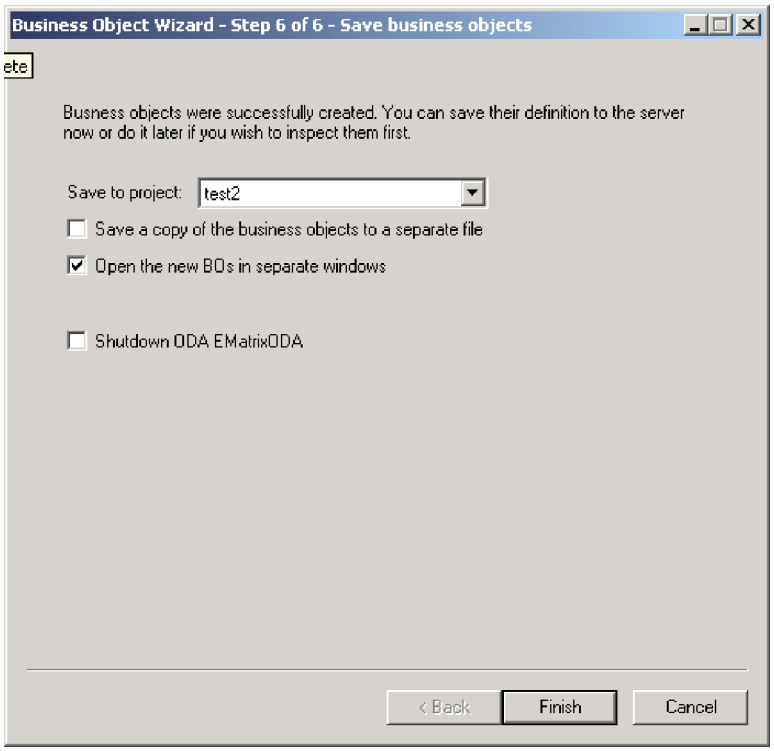

図 13. 「ビジネス・オブジェクトの保管」画面

ここで、ビジネス・オブジェクト定義を保管するロケーションを選択します。

生成されたビジネス・オブジェクト定義をオプションでファイルに保存できます。これを行うには、以下のステップを実行します。

1. 「**ビジネス・オブジェクトをファイルに保管**」にチェックマークを付けます。ダイアログ・ボックスが表示されます。
2. 保管する新規ビジネス・オブジェクト定義のコピーを格納する場所を入力します。

Business Object Designer Express によって、指定した場所にファイルが保管されます。

注: eMatrix 内で新規のカスタム・オブジェクトを作成した場合には、本章で説明されているように、ODA を使用してそのオブジェクトに新規のビジネス・オブジェクト定義を追加する必要があります。

ODA での作業が終了したら、「**ODA: eMatrix ODA をシャットダウン (Shutdown ODA: eMatrix ODA)**」にチェックマークを付け、「**完了**」をクリックします。

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。

InterChange Server Express の場合、ビジネス・オブジェクト定義ファイルをローカル・マシンに保管していて、このファイルをサーバーのリポジトリーにアップロードする必要がある場合は、「システム・インプリメンテーション・ガイド」を参照してください。

第 6 章 トラブルシューティングおよびエラー処理

この章では、アダプターが共通エラーを処理する方法について説明し、共通問題のトラブルシューティングについて説明します。

本章の内容は、次のとおりです。

- 41 ページの『エラー処理』
- 42 ページの『トラブルシューティング』

エラー処理

一般に、アダプターはエラーを検出すると常に、コネクター API によって、エラー・メッセージをログに記録します。

すべてのエラー・メッセージは、 17 ページの『ログ・ファイルとトレース・ファイルの使用』で説明したように、外部メッセージ・ファイルに保管されます。

すべての操作に関して、eMatrix への接続が確立できなかった場合、または処理中に接続が失われた場合、アダプターは、コード APPRESPONSETIMEOUT を戻し、致命的エラーをログに記録します。これにより、コネクターは終了し、電子メール通知が管理者に送信されます。

未解決のイベント

アダプターは、構成プロパティーの値に応じて、未解決または未完了のイベントからリカバリーするためのアクションを実行します。

表 14 にオプションがリストされています。

表 14. 未確定イベントの処理

「inDoubtEvents」の値	**実行するアクション**
FailOnStartUp	致命的エラーをログに記録し、終了します。
Reprocess	通常のイベントと同じように、イベントをブローカーにパブリッシュします。
LogError	イベントを無視し、エラーをログに記録します。
Ignore	イベントを無視します。

ポーリング・エラー

ポーリング処理でいくつかのエラーが発生する場合があります。 表 15 に最も一般的なものがリストされています。

表 15. ポーリング・エラーの処理

エラー	**アクション**
サブスクライブされていないため、イベントをパブリッシュできません。	エラー・メッセージをログに記録し、場合によりイベントを保存し、イベントの処理を続けます。

表 15. ポーリング・エラーの処理 (続き)

エラー	アクション
コネクター・コントローラーがアクティブでないため、イベントをパブリッシュできません。	イベント状況を 0 に設定し、SUCCESS コードを戻します。
エラーがこのイベントに固有のものです。例えば、イベントに無効な値が指定された場合などです。	エラー・メッセージをログに記録し、場合によりイベントを保存し、イベントの処理を続けます。
ブローカーは、エラー・メッセージを出してイベントを拒否します。	エラー・メッセージをログに記録し、場合によりイベントおよびエラー・メッセージを保存し、FAIL コードを戻します。
eMatrix と通信できません。その他のリカバリー不能エラーです。	致命エラー・メッセージをログに記録し、APPRESPONSETIMEOUT コードを戻します。

動詞処理エラー

アダプターは、ビジネス・オブジェクトを識別するために、固有のビジネス・オブジェクト ID または値の組み合わせ (タイプ、名前、改訂) を必要とします。指定された値が誤っている場合、または既存ビジネス・オブジェクトに一致しない場合、エラーが報告されます。

最も一般的な動詞処理エラーのいくつかを以下に示します。

- ビジネス・オブジェクトを作成する場合にアダプターが正常なときは、アダプターは、新しく作成したビジネス・オブジェクトを統合ブローカーに送り、値 SUCCEED を戻します。
  指定された値を持ったビジネス・オブジェクトがすでに存在する場合、アダプターは VALDUPES を戻し、新しいビジネス・オブジェクトを作成しません。
- ビジネス・オブジェクトを更新する場合にアダプターが正常なときは、アダプターは、新しく更新したビジネス・オブジェクトを統合ブローカーに送り、値 VALCHANGE を戻します。
- 処理中にエラーが発生する場合、アダプターはブローカーに FAIL コードを戻し、説明エラー・メッセージをログに記録します。

トラブルシューティング

このセクションでは、Adapter for eMatrix の使用時に発生する共通問題に関して、いくつかの簡単なトラブルシューティングの手法を説明します。

イベント通知

このセクションでは、イベント通知中に発生する可能性のある 4 つの共通問題について説明します。共通問題は以下のとおりです。

- イベント生成の失敗
- eMatrix イベント・ビジネス・オブジェクト名の誤り
- イベント・ボールトの未検出
- ビジネス・オブジェクト階層の誤り

イベント生成の検査

イベント通知機能をデバッグする場合、まず、イベントが正常に生成されているかどうかを検査します。これを行うには、以下のステップを実行します。

1. WBIEventLogger を呼び出すトリガーを起動するアクションを実行します。
2. 次に、以下のような照会を実行し、イベントが生成されたかどうかを確認します。

   ```
   temp query bus wbi_Event **;
   ```

3. イベントが生成されていない場合、WBIEventLogger を呼び出すトリガーがインストールされていることを確認します。
   eMatrix Business Modeler の該当するビジネス・オブジェクトのタイプまたは関係定義を開くことによって、トリガーがインストールされているかどうかを検査することができます。イベント通知トリガーのインストール例については、91 ページの『付録 C. サンプル・シナリオ』を参照してください。

   注: トリガーを呼び出すアクションを行うユーザーのユーザー名が、WBIEventLogger で指定された `adapterUser` 定数に一致する場合、イベントは生成されません。`adapterUser` 定数は、アダプターが eMatrix システムにログインするときに使用するユーザー名を指定します。
   そのようなイベントを無視する上で大事な点は、アダプターが eMatrix システムのエンティティーを変更することによって、イベントがログに記録されるようになることを防ぐことにあります。ここでは、アダプターはそれ自身の活動を認識しているため、それらをログに記録する必要がないことが前提になっています。テストを行う場合はこのことに注意してください。

4. WBIEventLogger を実行するトリガーが呼び出されているかどうかをテストする場合、matrix.ini ファイルに次のプロパティーを設定します。

   ```
   MX_MQL_TRACE=TRUE
   ```

   トリガーを呼び出すアクションを実行すると、次のようなメッセージが表示されます。

   ```
   18:15:00.826 MQL t@2900 Session: mx3f20221364f87ac7
   18:15:00.826 MQL t@2900 Program: WBIEventLogger
   18:15:00.826 MQL t@2900 args: -method recordEvent ${OBJECTID} n=wbi_cpu
   v=Update p=1
   18:15:08.417 MQL t@2900 End Program:MQL<3>
   ```

   このメッセージは、トリガーが、上にリストされた引数で、WBIEventLogger を実行したことを示しています。

eMatrix イベント・ビジネス・オブジェクト名の検査

イベントが生成されていることがわかっている場合でも、eMatrix アプリケーションをポーリングするたびにエラーが発生するときは、イベント・ビジネス・オブジェクトの名前を詳しく調べてください。ビジネス・オブジェクト名は、次のようになっている必要があります。

```
wbi_22032.41802.50936.705_1058897313350_1892833611
```

ビジネス・オブジェクト名に FROMOBJECT のような大文字がある場合、引数として無効な MARCO 値をトリガーに指定した可能性があります。これを解決するには、以下のようにします。

1. Business Modeler のすべてのビジネス・オブジェクト・タイプ定義を検査します。
2. それらが WBIEventLogger を呼び出すときに使用するトリガーを調べ、引数として指定した MARCO 値がすべて正しいことを確認します。

イベント・ボールトの検索

イベント・ポーリングのときに次のエラーを受け取る場合、アダプターが、コネクター・プロパティーで参照される EventVault を見つけられなかったことを意味します。エラーは次のとおりです。

```
System Error: #1600039: vault 'WBI_Events' does not exist
```

このエラーは、以下のいずれかの理由で発生します。

- プロパティーで誤ったボールトを参照している。
- ボールトが作成されていない。

InstallEventTables を実行して、ボールトが作成されていることを確認し、コネクター・プロパティーの EventVault の名前を検査してください。

ビジネス・オブジェクト階層の検査

イベント通知に使用されるビジネス・オブジェクト階層を作成する場合、階層に、親も eMatrix ビジネス・オブジェクト・タイプを表すような、eMatrix ビジネス・オブジェクト・タイプを表す子ビジネス・オブジェクト定義が含まれないようにしてください。

そのようなオブジェクトが存在すると、アダプターはポーリング時に失敗し、次のようなメッセージをログに記録します。

```
An error occurred while trying to validate your WBI business object
instance hierarchy.
```

このエラーは、eMatrix ビジネス・オブジェクトを表すビジネス・オブジェクトが、ヌルでない eMatrix ビジネス・オブジェクトも表す属性を含むことができないために発生します。

要求処理

このセクションでは、要求処理中に発生する可能性のある 2 つの共通問題について説明します。共通問題は以下のとおりです。

- コマンドにおけるストリング値の誤り
- 階層ビジネス・オブジェクトの値の誤り

コマンドにおけるストリングの処理

eMatrix コマンドを表すビジネス・オブジェクトの string 型属性の値は、コマンド処理前のコマンド・テンプレートへの挿入時に、引用符で囲まれません。したがって、引用符で囲まれていない空白文字を含む属性値は、2 語の MQL コマンドとして解釈されます。

ビジネス・オブジェクト階層の値

要求を送信する場合、階層内に、eMatrix ビジネス・オブジェクトを表す子があり、その親オブジェクトも eMatrix ビジネス・オブジェクトを表している場合があります。子ビジネス・オブジェクトがヌルでないことを確認してください。そのような階層が検出されると、アダプターはエラーをスローします。

一般的なヒント

2 つの一般的な問題およびそれらの解決法を以下に示します。

サポートされないビジネス・オブジェクト

ログまたはトレース・ファイルで、次のようなメッセージを受け取る場合があります。

Unable to find version “*.*.*” of business object definition “wbi_computer_bundle_to_computer”

これを解決するには、メッセージに示されたビジネス・オブジェクトを、Connector Configurator Express でサポートされるビジネス・オブジェクトのリストに追加する必要があります。

不完全なビジネス・オブジェクト関係

ODA を使用する場合、2 つのビジネス・オブジェクト・タイプとそれらの接続関係をモデル化したいときがあります。図 14 に示すように、必ず、階層関係にある 3 つすべてを選択してください。

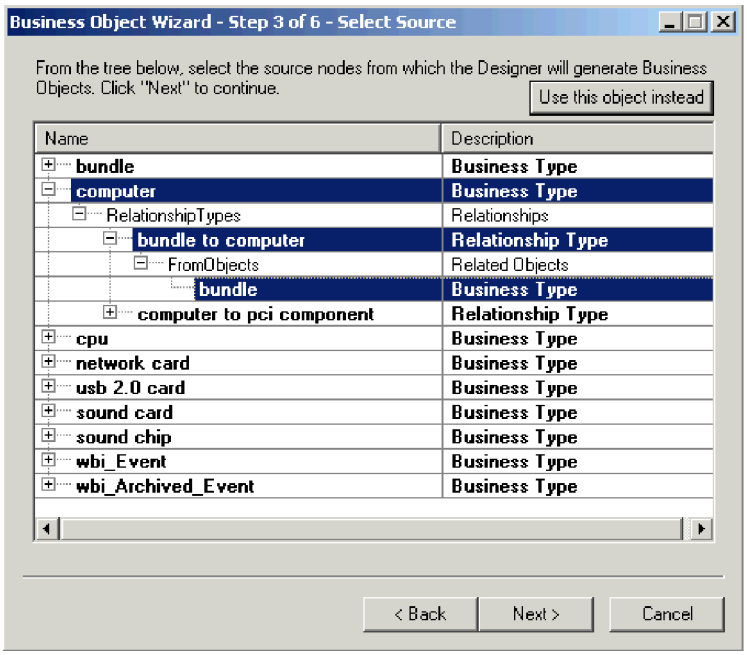

図 14. ビジネス・オブジェクト関係の選択

これを実行しないと、ODA は、タイプとそれらの共用関係が不完全なモデルを作成してしまいます。

付録 A. コネクターの標準構成プロパティー

この付録では、WebSphere Business Integration Server Express Adapter のコネクター・コンポーネントの標準構成プロパティーについて説明します。この付録の情報は、InterChange Server Express を対象としています。

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

新規プロパティー

以下の標準プロパティーは、本リリースで追加されました。

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

標準のコネクター・プロパティーの概要

コネクターには 2 つのタイプの構成プロパティーがあります。

- 標準構成プロパティー。フレームワークによって使用されます。
- アプリケーションまたはコネクター固有の構成プロパティー。エージェントによって使用されます。

これらのプロパティーで、アダプター・フレームワークおよびエージェントの実行時動作を決定します。

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

Connector Configurator Express の始動

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

Connector Configurator Express と System Manager は、Windows システム上でのみ動作します。

構成プロパティー値の概要

コネクターは、以下の順序に従ってプロパティーの値を決定します。

1. デフォルト

2. InterChange Server Express 統合ブローカー用のリポジトリー
3. ローカル構成ファイル
4. コマンド行

プロパティー・フィールドのデフォルトの長さは 255 文字です。 STRING プロパティー・タイプの長さに制限はありません。 INTEGER タイプの長さは、アダプターが稼働しているサーバーによって決定されます。

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

プロパティーの更新特性、つまりコネクター・プロパティーに対する変更がどのようにして有効になるかとその時期は、プロパティーの性質によって決まります。

標準コネクター・プロパティーには、以下の 4 種類の更新メソッドがあります。

- **動的**
  変更を System Manager に保管するとすぐに、新しい値が有効になります。ただし、コネクターがスタンドアロン・モードである (System Manager から独立している) 場合。
- **エージェント再始動 (InterChange Server Express のみ)**
  コネクター・エージェントを停止してから再始動した後にのみ、新しい値が有効になります。
- **コンポーネント再始動**
  System Manager でコネクターを停止してから再始動した後にのみ、新しい値が有効になります。エージェントやサーバー・プロセスを停止し再始動する必要はありません。
- **システム再始動**
  コネクター・エージェントおよびサーバーを停止してから再始動した後にのみ、新しい値が有効になります。

特定のプロパティーの更新方法を確認するには、「Connector Configurator Express」ウィンドウ内の「**更新メソッド**」列を参照するか、49 ページの表 16 の「更新メソッド」列を参照してください。

1 つの標準プロパティーを配置できる 3 つのロケーションがあります。一部のプロパティーは、複数のロケーションに存在することができます。

- **ReposController**
  プロパティーは、コネクター・コントローラー内に存在し、コネクター・コントローラーでのみ有効です。エージェント側で値を変更した場合、コントローラーには影響はありません。
- **ReposAgent**
  プロパティーは、エージェント内に存在し、エージェントでのみ有効です。プロパティーによっては、ローカル構成でこの値をオーバーライドすることができます。
- **LocalConfig**
  プロパティーは、コネクターの構成ファイル内に存在し、構成ファイルを介してのみ機能することができます。コントローラーが、プロパティーの値を変更する

ことはできず、システムを再配置してコントローラーを明示的に更新しない限り、構成ファイルに行われた変更を認識しません。

標準プロパティーの早見表

表 16 は、標準コネクター構成プロパティーの早見表です。すべてのコネクターがここに記載するすべてのプロパティーを必要とするわけではありません。また、プロパティーの設定が異なる場合があります。

各プロパティーの説明については、表の後のセクションを参照してください。

注: 表 16 の「注」の欄にある、「RepositoryDirectory を <REMOTE> に設定する」という句は、ブローカーが InterChange Server Express であることを示しています。

表 16. 標準構成プロパティーの要約

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
AdapterHelpName	有効な <Regional Setting> ディレクトリーを持つ <ProductDir>¥bin¥Data¥App¥Help の有効なサブディレクトリーの 1 つ	有効な場合はテンプレート名、あるいはブランク・フィールド	コンポーネント再始動	サポートされる地域の設定。chs_chn、cht_twn、deu_deu、esn_esp、fra_fra、ita_ita、jpn_jpn、kor_kor、ptb_bra、および enu_usa (デフォルト) など。
AdminInQueue	有効な JMS キュー名	*<CONNECTORNAME>*/ADMININQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合にのみ有効です。
AdminOutQueue	有効な JMS キュー名	*<CONNECTORNAME>*/ADMINOUTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合にのみ有効です。
AgentConnections	1 から 4	1	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が MQ または IDL で、Repository Directory の値を <REMOTE> に設定し、BrokerType の値が ICS である場合にのみ有効です。
AgentTraceLevel	0 から 5	0	ICS の場合は動的、それ以外の場合はコンポーネント再始動	
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	

表 16. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
CharacterEncoding	サポートされる任意のコード。リストに次のサブセットが表示されます。 ascii7、ascii8、SJIS、 Cp949、GBK、Big5、 Cp297、Cp273、Cp280、 Cp284、Cp037、Cp437	ascii7	コンポーネント再始動	このプロパティーは、C++ コネクターの場合のみ有効です。
CommonEventInfrastructure	true または false	false	コンポーネント再始動	
CommonEventInfrastructureURL	URL ストリング (例: corbaloc:iiop: host:2809)	デフォルト値なし。	コンポーネント再始動	このプロパティーは、CommonEvent Infrastructure の値が true である場合にのみ有効です。
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント再始動	このプロパティーは、RepositoryDirectory の値を <REMOTE> に設定し、BrokerType の値が ICS である場合にのみ有効です。
ContainerManagedEvents	ブランクまたは JMS	ブランク	コンポーネント再始動	このプロパティーは、Delivery Transport の値が JMS である場合にのみ有効です。
ControllerEventSequencing	true または false	true	動的	このプロパティーは、Repository Directory の値を <REMOTE> に設定し、BrokerType の値が ICS である場合にのみ有効です。
ControllerStoreAndForwardMode	true または false	true	動的	このプロパティーは、Repository Directory の値を <REMOTE> に設定し、BrokerType の値が ICS である場合にのみ有効です。
ControllerTraceLevel	0 から 5	0	動的	このプロパティーは、RepositoryDirectory の値を <REMOTE> に設定し、BrokerType の値が ICS である場合にのみ有効です。
DeliveryQueue	任意の有効な JMS キュー名	*<CONNECTORNAME>* /DELIVERYQUEUE	コンポーネント再始動	このプロパティーは、Delivery Transport の値が JMS である場合にのみ有効です。
DeliveryTransport	IDL または JMS	RepositoryDirectory の値が <REMOTE> である場合は IDL、それ以外の場合は JMS です。	コンポーネント再始動	RepositoryDirectory の値が <REMOTE> でない場合、このプロパティーの唯一の有効な値は JMS です。

表 16. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
DuplicateEventElimination	true または false	false	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合にのみ有効です。
EnableOidForFlowMonitoring	true または false	false	コンポーネント再始動	このプロパティーは、BrokerType の値が ICS である場合にのみ有効です。
FaultQueue	任意の有効なキュー名。	*<CONNECTORNAME>*/FAULTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合にのみ有効です。
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory、CxCommon.Messaging.jms.SonicMQFactory、または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合にのみ有効です。
jms.ListenerConcurrency	1 から 32767	1	コンポーネント再始動	このプロパティーは、jms.TransportOptimized の値が true である場合にのみ有効です。
jms.MessageBrokerName	FactoryClassName の値が IBM の場合は crossworlds.queue.manager を使用。	crossworlds.queue.manager	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合にのみ有効です。
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合にのみ有効です。
jms.Password	任意の有効なパスワード		コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS である場合にのみ有効です。
jms.TransportOptimized	true または false	false	コンポーネント再始動	このプロパティーは DeliveryTransport の値が JMS で、BrokerType の値が ICS である場合にのみ有効です。
jms.UserName	任意の有効な名前		コンポーネント再始動	このプロパティーは、Delivery Transport の値が JMS である場合にのみ有効です。
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	このプロパティーは、Repository Directory の値を <REMOTE> に設定し、BrokerType の値が ICS である場合にのみ有効です。

表 16. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
JvmMaxNativeStackSize	スタックのサイズ (キロバイト単位)	128k	コンポーネント再始動	このプロパティーは、Repository Directory の値を <REMOTE> に設定し、BrokerType の値が ICS である場合にのみ有効です。
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	このプロパティーは、Repository Directory の値を <REMOTE> に設定し、BrokerType の値が ICS である場合にのみ有効です。
ListenerConcurrency	1 から 100	1	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が MQ である場合にのみ有効です。
Locale	これは、サポートされるロケールの一部です。en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR	en_US	コンポーネント再始動	
LogAtInterchangeEnd	true または false	false	コンポーネント再始動	このプロパティーは、Repository Directory の値を <REMOTE> に設定し、BrokerType の値が ICS である場合にのみ有効です。
MaxEventCapacity	1 から 2147483647	2147483647	動的	このプロパティーは、Repository Directory の値を <REMOTE> に設定し、BrokerType の値が ICS である場合にのみ有効です。
MessageFileName	有効なファイル名	InterchangeSystem.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	*<CONNECTORNAME>* /MONITORQUEUE	コンポーネント再始動	このプロパティーは、DuplicateEventElimination の値が true で、ContainerManagedEvents に値がない場合にのみ有効です。
OADAutoRestartAgent	true または false	false	動的	このプロパティーは、Repository Directory の値を <REMOTE> に設定し、BrokerType の値が ICS である場合にのみ有効です。

表 16. 標準構成プロパティーの要約 (続き)

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

表 16. 標準構成プロパティーの要約 (続き)

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

標準のプロパティー

このセクションでは、標準コネクター構成プロパティーについて説明します。

AdapterHelpName

AdapterHelpName プロパティーは、コネクター固有の全般ヘルプ・ファイルが格納されているディレクトリーの名前です。このディレクトリーは、<*ProductDir*>¥bin¥Data¥App¥Help にあり、少なくとも言語ディレクトリー enu_usa

が格納されていることが必要です。このプロパティーには、ロケールに応じてその他のディレクトリーが含まれていることがあります。

デフォルト値はテンプレート名です (テンプレート名が有効な場合)。あるいは、ブランクです。

AdminInQueue

AdminInQueue プロパティーは、統合ブローカーがコネクターに管理メッセージを送信する際に使用するキューを指定します。

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

AdminOutQueue

AdminOutQueue プロパティーは、コネクター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を指定します。

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

AgentConnections

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

このプロパティーのデフォルト値は 1 です。

AgentTraceLevel

AgentTraceLevel プロパティーは、アプリケーション固有のコンポーネントのトレース・メッセージのレベルを設定します。コネクターは、設定されたトレース・レベル以下の該当するトレース・メッセージをすべてデリバリーします。

デフォルト値は 0 です。

ApplicationName

ApplicationName プロパティーは、コネクター・アプリケーションの名前を一意的に識別します。この名前は、システム管理者が統合環境をモニターするために使用します。コネクターを実行する前に、このプロパティーに値を指定する必要があります。

デフォルトは、コネクターの名前です。

BrokerType

BrokerType プロパティーは、使用する統合ブローカー・タイプを示します。値は ICS (InterChange Server Express) です。

CharacterEncoding

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

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

デフォルトでは、サポートされる文字エンコードの一部のみが表示されます。リストに、サポートされる他の値を追加するには、製品ディレクトリー (*<ProductDir>*) にある ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。詳細については、本書の付録『Connector Configurator Express』を参照してください。

ConcurrentEventTriggeredFlows

ConcurrentEventTriggeredFlows プロパティーは、コネクターがイベントのデリバリー時に並行処理できるビジネス・オブジェクトの数を決定します。この属性の値を、並行してマップおよびデリバリーされるビジネス・オブジェクトの数に設定します。例えば、このプロパティーの値を 5 に設定すると、5 個のビジネス・オブジェクトが並行して処理されます。

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

ソース・アプリケーションから宛先アプリケーションまでのフロー全体に並行処理を実装するには、次のプロパティーを構成する必要があります。

- Maximum number of concurrent events プロパティーの値を複数のスレッドを使用できる値に増やすことによって、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処理できるように構成します。

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。

このプロパティーは、RepositoryDirectory プロパティーを <REMOTE> に設定している場合にのみ有効です。

デフォルト値は 1 です。

ContainerManagedEvents

ContainerManagedEvents プロパティーでは、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、1 つの JMS トランザクションとして宛先キューに配置されます。

このプロパティーを JMS に設定する場合は、次のプロパティーも同時に設定して、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。

- PollQuantity = 1 から 500

- SourceQueue = /SOURCEQUEUE

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

これらのプロパティーはアプリケーションに固有ですが、次に示すようないくつかのサンプル値があります。

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

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

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

ContainerManagedEvents プロパティーは、 DeliveryTransport プロパティーの値を JMS に設定している場合にのみ有効です。

デフォルト値はありません。

ControllerEventSequencing

ControllerEventSequencing プロパティーは、コネクター・コントローラーでのイベントの順序付けを可能にします。

このプロパティーは、RepositoryDirectory プロパティーの値を <REMOTE> に設定している (BrokerType が ICS) 場合にのみ有効です。

デフォルト値は true です。

ControllerStoreAndForwardMode

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

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

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

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

このプロパティーは、RepositoryDirectory プロパティーの値を <REMOTE> に設定している (BrokerType プロパティーの値が ICS) 場合にのみ有効です。

デフォルト値は true です。

ControllerTraceLevel

ControllerTraceLevel プロパティーは、コネクター・コントローラーのトレース・メッセージのレベルを設定します。

このプロパティーは、RepositoryDirectory プロパティーの値を <REMOTE> に設定している場合にのみ有効です。

デフォルト値は 0 です。

DeliveryQueue

DeliveryQueue プロパティーは、コネクターから統合ブローカーへビジネス・オブジェクトが送信されるときに使用されるキューを定義します。

このプロパティーは、DeliveryTransport プロパティーの値を JMS に設定している場合にのみ有効です。

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

DeliveryTransport

DeliveryTransport プロパティーは、イベントのデリバリーのためのトランスポート機構を指定します。Java Messaging Service の場合、値は JMS です。

- RepositoryDirectory プロパティーを <REMOTE> に設定した場合、DeliveryTransport プロパティーの値を IDL または JMS に指定することができます。デフォルトは IDL です。
- RepositoryDirectory プロパティーの値がローカル・ディレクトリーである場合、指定可能な値は JMS のみです。

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

デフォルト値は JMS です。

JMS

JMS トランスポート機構は、Java Messaging Service (JMS) による、コネクターとクライアント・コネクター・フレームワーク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、Connector Configurator Express に、jms.MessageBrokerName、jms.FactoryClassName、jms.Password、およ

び jms.UserName などの追加の JMS プロパティーが表示されます。
jms.MessageBrokerName および jms.FactoryClassName は、このトランスポートの必須プロパティーです。

InterChange Server Espress (ICS) が統合ブローカーであるときに、以下の環境でコネクターに JMS トランスポート機構を使用すると、メモリーに制限がある場合があります。

この環境では、WebSphere MQ クライアント内でメモリーが使用されるため、(サーバー側の) コネクター・コントローラーと (クライアント側の) コネクターの両方を始動するのは困難な場合があります。ご使用のシステムのプロセス・ヒープ・サイズが 768 MB 未満である場合には、次の変数およびプロパティーを設定してください。

- CWSharedEnv.sh スクリプトに LDR_CNTRL 環境変数を設定します。

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

  ```
  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 に設定します。このプロパティーの詳細については、「*WebSphere Business Integration Server Express インストール・ガイド (Windows 版)*」を参照してください。

DuplicateEventElimination

このプロパティーの値が true である場合、JMS 対応コネクターがデリバリー・キューに重複イベントをデリバリーしないように設定できます。この機能を使用するには、コネクターの開発時に、コネクターに、アプリケーション固有のコード内でビジネス・オブジェクトの ObjectEventId 属性として一意のイベント ID が設定されている必要があります。

注: このプロパティーの値が true である場合、保証付きイベント・デリバリーを提供できるように、MonitorQueue プロパティーを使用可能に設定する必要があります。

デフォルト値は false です。

EnableOidForFlowMonitoring

このプロパティーが true である場合、アダプター・ランタイムにより、フロー・モニター用に着信 ObjectEventID が外部キーとしてマークされます。

このプロパティーは、BrokerType プロパティーを ICS に設定した場合にのみ有効です。

デフォルト値は false です。

FaultQueue

メッセージの処理中にコネクターで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の説明とともに FaultQueue プロパティーに指定されたキューに移動します。

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

jms.FactoryClassName

jms.FactoryClassName プロパティーは、JMS プロバイダーのためにインスタンスを生成するクラス名を指定します。 DeliveryTransport プロパティーの値が JMS である場合は、このプロパティーを設定する必要があります。

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

jms.ListenerConcurrency

jms.ListenerConcurrency プロパティーは、JMS コントローラーの並行リスナーの数を指定します。このプロパティーは、コントローラー内でメッセージを並行して取り出し、処理するスレッドの数を指定します。

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

デフォルト値は 1 です。

jms.MessageBrokerName

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

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

例えば、次のようになります。

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

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

jms.NumConcurrentRequests

jms.NumConcurrentRequests プロパティーは、コネクターに対して同時に送信することができる並行サービス呼び出し要求の数 (最大値) を指定します。この最大値に達した場合、新規のサービス呼び出しがブロックされ、別の要求が完了するのを待ってから続行する必要があります。

デフォルト値は 10 です。

jms.Password

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

デフォルト値はありません。

jms.TransportOptimized

jms.TransportOptimized プロパティーは、 WIP (処理中の作業) を最適化するかどうかを判別します。WIP を最適化するには、WebSphere MQ プロバイダーが必要です。最適化された WIP を処理するために、メッセージング・プロバイダーが以下の処理を実行できなければなりません。

1. メッセージをキューから除去せずに読み取る。
2. 特定の ID のメッセージ全体を受信側のメモリー・スペースに転送せずに削除する。
3. 特定の ID を使用してメッセージを読み取る (リカバリー目的で必要)。
4. 読み取られていないイベントの発生時点を追跡する。

JMS API は上の条件 2 および 4 を満たさないため、最適化された WIP 用に使用できませんが、MQ Java API は上の 4 つの条件すべてを満たすため、最適化された WIP に必要です。

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

デフォルト値は false です。

jms.UserName

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

デフォルト値はありません。

JvmMaxHeapSize

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

このプロパティーは、RepositoryDirectory プロパティーの値を <REMOTE> に設定している場合にのみ有効です。

デフォルト値は 128M です。

JvmMaxNativeStackSize

JvmMaxNativeStackSize プロパティーは、エージェントの最大ネイティブ・スタック・サイズ (キロバイト単位) を指定します。

このプロパティーは、RepositoryDirectory プロパティーの値を <REMOTE> に設定している場合にのみ有効です。

デフォルト値は 128K です。

JvmMinHeapSize

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

このプロパティーは、RepositoryDirectory プロパティーの値を <REMOTE> に設定している場合にのみ有効です。

デフォルト値は 1M です。

ListenerConcurrency

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

このプロパティーは、MQ トランスポートを使用するコネクターでのみ有効です。DeliveryTransport プロパティーの値には、MQ を指定してください。

デフォルト値は 1 です。

Locale

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

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、以下のように説明されます。
ll は 2 文字の言語コード (小文字) です。
TT は 2 文字の国または地域コード (大文字) です。
codeset は、関連する文字コード・セットの名前 (オプション) です。

デフォルトでは、サポートされるロケールの一部のみがリストされます。サポートされるその他の値をリストに追加するには、*<ProductDir>*¥bin ディレクトリーにある ¥Data¥Std¥stdConnProps.xml ファイルを変更します。詳細については、本書の付録『Connector Configurator Express』を参照してください。

コネクターが国際化されていない場合、このプロパティーの有効な値は en_US のみです。特定のコネクターがグローバル化に対応しているかどうかを判別するには、そのアダプターのユーザーズ・ガイドを参照してください。

デフォルト値は en_US です。

LogAtInterchangeEnd

LogAtInterchangeEnd プロパティーは、エラーを統合ブローカーのログの宛先に記録するかどうかを指定します。

ログ宛先にログを記録すると、電子メール通知もオンになります。これにより、エラーまたは致命的エラーが発生すると、InterchangeSystem.cfg ファイルに MESSAGE_RECIPIENT の値として指定された受信側に対して電子メール・メッセージが生成されます。例えば、LogAtInterChangeEnd の値が true である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の受信側に、電子メール・メッセージが送信されます。

このプロパティーは、RespositoryDirectory プロパティーを <REMOTE> に設定している場合 (BrokerType の値が ICS) にのみ有効です。

デフォルト値は false です。

MaxEventCapacity

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

このプロパティーは、RespositoryDirectory プロパティーを <REMOTE> に設定している場合 (BrokerType の値が ICS) にのみ有効です。

値は 1 から 2147483647 の間の正整数です。

デフォルト値は 2147483647 です。

MessageFileName

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

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

注: コネクターについて、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。

デフォルト値は InterchangeSystem.txt です。

MonitorQueue

MonitorQueue プロパティーは、コネクターが重複イベントをモニターするために使用する論理キューを指定します。

これは、DeliveryTransport プロパティーの値が JMS で、DuplicateEventElimination の値が true である場合にのみ有効です。

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

OADAutoRestartAgent

OADAutoRestartAgent プロパティーは、コネクターが自動再始動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 WebSphere MQ-triggered 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。WebSphere MQ により起動される OAD 機能の構成方法については、「*WebSphere Business Integration Server Express インストール・ガイド (Windows 版)*」を参照してください。

このプロパティーは、RespositoryDirectory プロパティーを <REMOTE> に設定している場合 (BrokerType の値が ICS) にのみ有効です。

デフォルト値は false です。

OADMaxNumRetry

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

このプロパティーは、RespositoryDirectory プロパティーを <REMOTE> に設定している場合 (BrokerType の値が ICS) にのみ有効です。

デフォルト値は 1000 です。

OADRetryTimeInterval

OADRetryTimeInterval プロパティーは、MQ により起動される WebSphere MQ-triggered Object Activation Daemon (OAD) の再試行時間間隔の分数を指定します。コネクター・エージェントがこの再試行時間間隔内に再始動しない場合は、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。

このプロパティーは、RespositoryDirectory プロパティーを <REMOTE> に設定している場合 (BrokerType の値が ICS) にのみ有効です。

デフォルト値は 10 です。

PollEndTime

PollEndTime プロパティーは、イベント・キューのポーリングを停止する時刻を指定します。形式は *HH:MM* です。ここで、*HH* は 0 から 23 時を表し、*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、値なしの HH:MM です。この値は必ず変更する必要があります。

アダプター・ランタイムが次のいずれかの状態を検出した場合、

- PollStartTime は設定されているが、PollEndTime は設定されていない
- PollEndTime は設定されているが、PollStartTime は設定されていない

PollFrequency プロパティー用に構成された値を使用して、ポーリングが行われます。

PollFrequency

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

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

このプロパティーでは、次の値が有効です。

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

デフォルト値は 10000 です。

重要: 一部のコネクターでは、このプロパティーの使用が制限されています。このようなコネクターが存在する場合には、アダプターのインストールと構成に関する章で制約事項が説明されています。

PollQuantity

PollQuantity プロパティーは、コネクターがアプリケーションからポーリングする項目の数を指定します。アダプターにポーリング数を設定するコネクター固有のプロ

パティーがある場合、このコネクター固有のプロパティーの設定値により標準プロパティーの値がオーバーライドされます。

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

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

- コネクターは、1 度ポーリングを受けると、添付として読み取られるメッセージの本文を検出します。この MIME タイプにはデータ・ハンドラーが指定されていないため、コネクターはメッセージを無視します。
- コネクターは、最初の BO の添付を処理します。データ・ハンドラーは、この MIME タイプで使用できるため、Visual Test Connector にビジネス・オブジェクトを送信します。
- コネクターは 2 度目のポーリングを受けると、2 番目の BO の添付を処理します。データ・ハンドラーは、この MIME タイプで使用できるため、Visual Test Connector にビジネス・オブジェクトを送信します。
- これが受け入れられると、3 番目の BO の添付が送信されます。

PollStartTime

PollStartTime プロパティーは、イベント・キューのポーリングを開始する時刻を指定します。形式は *HH:MM* です。ここで、*HH* は 0 から 23 時を表し、*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、値なしの HH:MM です。この値は必ず変更する必要があります。

アダプター・ランタイムが次のいずれかの状態を検出した場合、

- PollStartTime は設定されているが、PollEndTime は設定されていない
- PollEndTime は設定されているが、PollStartTime は設定されていない

PollFrequency プロパティー用に構成された値を使用して、ポーリングが行われます。

RepositoryDirectory

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

統合ブローカーが ICS である場合、コネクターがこの情報を InterChange Server Express リポジトリーから取得するため、この値を <REMOTE> に設定する必要があります。

統合ブローカーが WebSphere Message Broker または WAS である場合は、この値をデフォルトで<*ProductDir*>¥repository に設定する必要があります。ただし、任意の有効なディレクトリー名を設定することもできます。

RequestQueue

RequestQueue プロパティーは、統合ブローカーが、ビジネス・オブジェクトをコネクターに送信するときに使用されるキューを指定します。

このプロパティーは、DeliveryTransport プロパティーが JMS である場合にのみ有効です。

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

ResponseQueue

ResponseQueue プロパティーは、JMS 応答キューを指定します。JMS 応答キューは、応答メッセージをコネクター・フレームワークから統合ブローカーへデリバリーします。統合ブローカーが InterChange Server Express (ICS) 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。

このプロパティーは、DeliveryTransport プロパティーが JMS である場合にのみ有効です。

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

RestartRetryCount

RestartRetryCount プロパティーは、コネクターによるコネクター自体の再始動の試行回数を指定します。このプロパティーを並列に接続されるコネクターに対して使用する場合、コネクターのマスター側のアプリケーション固有のコンポーネントがクライアント側のアプリケーション固有のコンポーネントを再始動しようとする回数が指定されます。

デフォルト値は 3 です。

RestartRetryInterval

RestartRetryInterval プロパティーは、コネクターによるコネクター自体の再始動の試行間隔を分単位で指定します。このプロパティーを、並列にリンクされるコネクターに対して使用する場合、コネクターのマスター側のアプリケーション固有のコンポーネントが、クライアント側のアプリケーション固有のコンポーネントを再始動しようとする間隔が指定されます。

このプロパティーに指定可能な値は、1 から 2147483647 の範囲です。

デフォルト値は 1 です。

RHF2MessageDomain

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

以下にヘッダーの例を示します。

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

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

指定可能な値は mrm と xml です。デフォルト値は mrm です。

SourceQueue

SourceQueue プロパティーは、JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、56 ページの『ContainerManagedEvents』を参照してください。

このプロパティーは、DeliveryTransport の値が JMS で、ContainerManagedEvents の値を指定している場合にのみ有効です。

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

SynchronousRequestQueue

SynchronousRequestQueue プロパティーは、同期応答を要求する要求メッセージを、コネクター・フレームワークからブローカーに配信します。このキューは、コネクターが同期実行を使用する場合にのみ必要です。同期実行の場合、コネクター・フレームワークは、同期要求キューにメッセージを送信し、同期要求キューでブローカーの応答を待機します。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 と一致する相関 ID が含まれています。

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

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

SynchronousRequestTimeout

SynchronousRequestTimeout プロパティーは、コネクターが同期要求への応答を待機する時間をミリ秒単位で指定します。コネクターは、指定された時間内に応答を受信しなかった場合、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。

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

デフォルト値は 0 です。

SynchronousResponseQueue

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

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

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

TivoliMonitorTransactionPerformance

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

デフォルト値は `false` です。

WireFormat

WireFormat プロパティーは、トランスポートに関するメッセージ形式を指定します。

- RepositoryDirectory プロパティーの値がローカル・ディレクトリーである場合、値は `CwXML` になります。
- RepositoryDirectory プロパティーの値がリモート・ディレクトリーである場合、値は `CwBO` になります。

付録 B. Connector Configurator Express

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

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

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

この付録では、次のトピックについて説明します。

- 『Connector Configurator Express の概要』
- 73 ページの『コネクター固有のプロパティー・テンプレートの作成』
- 76 ページの『新しい構成ファイルの作成』
- 80 ページの『構成ファイル・プロパティーの設定』

Connector Configurator Express の概要

Connector Configurator Express では、InterChange Server Express 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。

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

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成します。
- **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、InterChange Server Express 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。

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

標準プロパティーはすべてのコネクターにより使用されるので、標準プロパティーを新規に定義する必要はありません。ファイルを作成すると、Connector Configurator Express 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 Express で各標準プロパティーの値を設定する必要があります。

標準プロパティーの範囲は、ブローカーと構成によって異なる可能性があります。特定のプロパティーに特定の値が設定されている場合にのみ使用できるプロパティーがあります。Connector Configurator Express の「標準のプロパティー」ウィンドウには、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、73 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

Connector Configurator Express の始動

以下の 2 種類のモードで Connector Configurator Express 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、System Manager を実行せずに Connector Configurator Express を実行し、コネクター構成ファイルを編集できます。

これを行うには、以下のステップを実行します。

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

Connector Configurator Express を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (79 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

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

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

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

4. 「**システム接続: Integration Broker**」の隣のプルダウン・メニューをクリックして、ICS を選択します。

既存の構成ファイルを編集するには、以下のステップを実行します。

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

コネクター固有のプロパティー・テンプレートの作成

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

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

- テンプレートの新規作成については、73 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でこのテンプレートを保管します。既存のテンプレートは ¥ProductDir¥bin¥Data¥App ディレクトリーにあります。

新規テンプレートの作成

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

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

1. 「**ファイル**」>「**新規**」>「**コネクター固有プロパティー・テンプレート**」をクリックします。
2. 「**コネクター固有プロパティー・テンプレート**」 ダイアログ・ボックスが表示されます。
   - 「**新規テンプレート名を入力してください**」の下の「**名前**」フィールドに、新規テンプレートの名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
   - テンプレートに含まれているコネクター固有のプロパティー定義を調べるには、「**テンプレート名**」表示でそのテンプレートの名前を選択します。そのテンプレートに含まれているプロパティー定義のリストが「**テンプレートのプレビュー**」表示に表示されます。

3. テンプレートを作成するときには、ご使用のコネクターに必要なプロパティー定義に類似したプロパティー定義が含まれている既存のテンプレートを使用できます。ご使用のコネクターで使用するコネクター固有のプロパティーが表示されるテンプレートが見つからない場合は、自分で作成する必要があります。
   - 既存のテンプレートを変更する場合には、「**変更する既存のテンプレートを選択してください: 検索テンプレート**」の下の「**テンプレート名**」テーブルのリストから、テンプレート名を選択します。
   - このテーブルには、現在使用可能なすべてのテンプレートの名前が表示されます。テンプレートを検索することもできます。

一般特性の指定

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

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

「**プロパティー・サブタイプ**」は、「**プロパティー・タイプ**」が「String」のときに選択することができます。これは、構成ファイルを保管するときに構文検査を行うオプションの値です。デフォルトは、ブランク・スペースで、プロパティーにサブタイプがなかったことを示します。

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

値の指定

「**値**」タブを使用すると、プロパティーの最大長、最大複数値、デフォルト値、または値の範囲を設定できます。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**」のフィールドに値を入力します。

新規プロパティー値を作成するには、以下のステップを実行します。

1. 「値」列見出しの左側にある四角を右マウス・ボタンでクリックします。
2. ポップアップ・メニューから、「**追加**」を選択して、「プロパティー値」ダイアログ・ボックスを表示します。プロパティーのタイプによって、ダイアログ・ボックスで値のみ、または値とその範囲の両方を入力することができます。

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

「**値**」パネルには、3 つの列からなるテーブルが表示されます。

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

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

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

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

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

依存関係の設定

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

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

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

   == (等しい)

   != (等しくない)

   > (より大)

   < (より小)

   >= (より大か等しい)

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

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

パス名の設定

パス名を設定する際の一般規則は以下のとおりです。

- Windows のファイル名の最大長は 255 文字です。
- Windows では、絶対パス名を [Drive:][Directory]¥filename の形式 (例: C:¥WebSphereAdapters¥bin¥Data¥Std¥StdConnProps.xml) で指定する必要があります。
- キュー名には、先行スペースまたは組み込みスペースを指定できません。

新しい構成ファイルの作成

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

また、ファイルの拡張検証のためのオペレーティング・システムを選択します。ツールバーに、「**ターゲット・システム**」と呼ばれるドロップ・リストがあります。これを使用して、プロパティーの拡張検証用のターゲット・オペレーティング・システムを選択することができます。 Windows で使用可能なオプションは、拡張検証なし (拡張検証をオフにする) です。始動時のデフォルトは Windows です。

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

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

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

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

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

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

- 「**Integration Broker**」フィールドで、ICS を選択します。
- この章で後述する説明に従って、「**新規コネクター**」ウィンドウの残りのフィールドに値を入力します。

コネクター固有のテンプレートからの構成ファイルの作成

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

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

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

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

     「ICS」をクリックします。
   - **コネクター固有プロパティー・テンプレートを選択 (Select Connector-Specific Property Template)**

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

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

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

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

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

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

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

既存ファイルを使用してコネクターを構成するには、Connector Configurator Express でそのファイルを開き、構成を修正し、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 Express 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 Express を始動します。
3. 「**ファイル**」>「**開く**」>「**プロジェクトから**」をクリックします。

構成ファイルの完成

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

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

1. 「**標準のプロパティー**」タブで、BrokerType プロパティーの値フィールドを選択します。ドロップダウン・メニューで、値 ICS を選択します。
2. 選択したブローカーに関連付けられているコネクター・プロパティーが「標準のプロパティー」タブに表示されます。テーブルに、「**プロパティー名**」、「**値**」、「**タイプ**」、「**サブタイプ**」(「タイプ」が「String」の場合)、「**説明**」、および「**更新メソッド**」が表示されます。
3. ここでファイルを保管するか、または 83 ページの『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。
4. 構成が完了したら、「**ファイル**」>「**保管**」>「**プロジェクトに**」を選択するか、または「**ファイル**」>「**保管**」>「**ファイルに**」を選択します。

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

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

   構成ファイルを作成する前に、「**ターゲット・システム**」ドロップ・リストを使用して、プロパティーの拡張検証用のターゲット・オペレーティング・システムを選択しています。

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

「**ターゲット・システム**」ドロップ・リストから「Windows」の値を選択して拡張検証機能の使用を選択した場合、システムがタイプに加えてプロパティー・サブタイプを検証し、検証に失敗すると警告メッセージを表示します。

構成ファイル・プロパティーの設定

新規のコネクター構成ファイルを作成して名前を付けるとき、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 Express 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 Express では、すべてのブローカーで実行されているコネクターで、以下のカテゴリーのプロパティーに値が設定されている必要があります。

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

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

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

- 関連付けられたマップ
- セキュリティー

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

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

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

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

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

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下のステップを実行します。

1. 値を設定するフィールド内でクリックします。
2. 値を入力するか、ドロップダウン・メニューが表示された場合にはメニューから値を選択します。

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

特定の標準プロパティーについての詳細情報を取得するには、「標準のプロパティー」タブの付いたシートの中のそのプロパティーの「説明」列の項目を左マウス・ボタンでクリックします。全般ヘルプをインストールしている場合は、右側に矢印ボタンが表示されます。このボタンをクリックすると、ヘルプ・ウィンドウが開き、標準のプロパティーの詳細が表示されます。

注: ホット・ボタンが表示されない場合は、そのプロパティーに対する全般ヘルプが検出されなかったということです。

全般ヘルプをインストールしている場合、そのファイルは
<*ProductDir*>¥bin¥Data¥Std¥Help¥<*RegionalSetting*>¥にあります。

コネクター固有の構成プロパティーの設定

コネクター固有の構成プロパティーの場合、プロパティー名の追加または変更、値の構成、プロパティーの削除、およびプロパティーの暗号化が可能です。プロパティーのデフォルトの長さは 255 文字です。

1. グリッドの左上端の部分で右マウス・ボタンをクリックします。ポップアップ・メニュー・バーが表示されます。プロパティーを追加するときは「**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。

   注: プロパティーの「タイプ」が「String」の場合、「サブタイプ」ドロップ・リストからサブタイプを選択することができます。このサブタイプは、プロパティーの拡張検証に使用されます。
3. プロパティーを暗号化するには、「**暗号化**」ボックスを選択します。
4. 特定のプロパティーについての詳細情報を取得するには、そのプロパティーの「説明」列の中の項目を左マウス・ボタンでクリックします。全般ヘルプをインストールしている場合は、ホット・ボタンが表示されます。ホット・ボタンをクリックすると、ヘルプ・ウィンドウが開き、標準のプロパティーの詳細が表示されます。

   注: ホット・ボタンが表示されない場合は、そのプロパティーに対する全般ヘルプが検出されなかったということです。
5. 81 ページの『標準コネクター・プロパティーの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

全般ヘルプ・ファイルがインストールされていて、AdapterHelpName プロパティーがブランクの場合、Connector Configurator Express は <*ProductDir*>¥bin¥Data¥App¥Help¥<*RegionalSetting*>¥ にあるアダプター固有の全般ヘルプ・ファイルを指示します。それ以外の場合、Connector Configurator Express は、 <*ProductDir*>¥bin¥Data¥App¥Help¥<*AdapterHelpName*>¥<*RegionalSetting*>¥ にある、アダプター固有の全般ヘルプ・ファイルを指示します。付録 A『コネクターの標準構成プロパティー』の『標準のプロパティー』で説明している AdapterHelpName プロパティーを参照してください。

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

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

コネクター・プロパティーの暗号化

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

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

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

更新メソッド

付録 A『コネクターの標準構成プロパティー』の 47 ページの『標準のコネクター・プロパティーの概要』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 Express の「**サポートされているビジネス・オブジェクト**」タブを使用します。汎用ビジネス・オブジェクトと、アプリケーション固有のビジネス・オブジェクトの両方を指定する必要があり、またそれらのビジネス・オブジェクト間のマップの関連を指定することが必要です。

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

ブローカーとしての InterChange Server Express

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、以下のステップを実行します。

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。ドロップ・リストが表示され、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します。
2. 追加するビジネス・オブジェクトをクリックします。
3. ビジネス・オブジェクトの「**エージェント・サポート**」(以下で説明) を設定します。

4. 「Connector Configurator Express」ウィンドウの「ファイル」メニューで、「**プロジェクトに保管**」をクリックします。追加したビジネス・オブジェクト定義に指定されたサポートを含む、変更されたコネクター定義が、System Manager の ICL (Integration Component Library) プロジェクトに保管されます。

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

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

サポートされるリストからビジネス・オブジェクトを削除すると、コネクター定義が変更され、削除されたビジネス・オブジェクトはコネクターのこのインプリメンテーションで使用不可になります。コネクターのコードに影響したり、そのビジネス・オブジェクト定義そのものが System Manager から削除されることはありません。

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

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

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

関連付けられたマップ

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

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

ジェクトに変換したり、汎用ビジネス・オブジェクトをアプリケーション固有のビジネス・オブジェクトに変換したりするときに、どのマップを使用するかが決定されます。

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

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

「**関連付けられたマップ**」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的バインディング**

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

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

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

  以下のをステップを実行して、マップを明示的にバインドします。

  1. 「**明示的 (Explicit)**」列で、バインドするマップのチェック・ボックスにチェックマークを付けます。
  2. ビジネス・オブジェクトに関連付けるマップを選択します。

3. 「Connector Configurator Express」ウィンドウの「**ファイル**」メニューで、「**プロジェクトに保管**」をクリックします。
4. プロジェクトを InterChange Server Express に配置します。
5. 変更を有効にするため、サーバーをリブートします。

セキュリティー

Connector Configurator Express の「**セキュリティー**」タブを使用して、メッセージのさまざまなプライバシー・レベルを設定することができます。この機能は、DeliveryTransport プロパティーを JMS に設定しているときに限って使用できます。

デフォルトでは、「プライバシー」はオフになります。これを使用可能にするには、「**プライバシー**」ボックスをチェックしてください。

「**鍵ストア・ターゲット・システムの絶対パス名**」は次のようになります。

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

このパスおよびファイルは、コネクターを始動させる予定のシステム、つまりターゲット・システム上に存在している必要があります。

ターゲット・システムが稼働中の唯一のシステムである場合にのみ、右側にある「参照」ボタンを使用することができます。「**プライバシー**」が使用可能で、メニュー・バーの 「**ターゲット・システム**」が「Windows」に設定されていない場合は、このボタンはグレーで表示されます。

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

- “”: デフォルトです。メッセージ・カテゴリーに対して、プライバシー・レベルが設定されていないときに使用されます。
- none: デフォルトと同じではありません。メッセージ・カテゴリーに対して意図的にプライバシー・レベル「なし」を設定するときに、これを使用します。
- integrity
- privacy
- integrity_plus_privacy

「**鍵の保守**」機能では、サーバーおよびアダプター用のインポートおよびエクスポート公開鍵を生成することができます。

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

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

- エクスポート証明書は、デフォルトでは鍵ストアと同じ値になります。ただし、ファイル拡張子は <filename>.cer となります。

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

- インポート証明書は、デフォルトでは *<ProductDir>*¥bin¥ics.cer (ファイルがシステム上に存在する場合) となります。
- インポートの「証明書関連」はサーバー名である必要があります。サーバーが登録されている場合は、ドロップ・リストからサーバーを選択できます。

DeliveryTransport の値が IDL である場合にのみ、「**アダプター・アクセス制御**」機能が使用可能です。デフォルトでは、アダプターはゲスト ID でログインします。「**ゲスト ID の使用**」ボックスをチェックしない場合、「**アダプター ID**」フィールドおよび「**アダプター・パスワード**」フィールドが使用可能になります。

トレース/ログ・ファイル値の設定

コネクター構成ファイルまたはコネクター定義ファイルを開くと、Connector Configurator Express は、そのファイル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 Express 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、以下のステップを実行します。

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: ロギング・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『コネクターの標準構成プロパティー』の『標準のプロパティー』にある ContainerManagedEvents の下の説明を参照してください。

構成ファイルの保管

コネクターの構成が完了したら、コネクター構成ファイルを保管します。Connector Configurator Express では、構成中に選択したブローカー・モードで構成ファイルを保管します。 Connector Configurator Express のタイトル・バーには InterChange Server Express が現在使用しているブローカー・モードが常に表示されます。

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

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

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

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、以下のステップを実行します (オプション)。

- Connector Configurator Express 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・ウィンドウの使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関連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 Express の使用

Connector Configurator Express 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 Express では、ネイティブなエンコード方式を使用しています。構成ファイルに書き込む場合は UTF-8 エンコード方式を使用します。

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

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

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

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

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

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

付録 C. サンプル・シナリオ

この付録では、eMatrix の有効なサンプル・シナリオについて説明します。このサンプル・シナリオは、Adapter for eMatrix を使用して、eMatrix を、WebSphere Business Integration アダプターがサポートする他のアプリケーションおよびテクノロジーと統合する方法を示します。

この付録では、次のトピックについて説明します。

- 91 ページの『概要』
- 91 ページの『前提事項』
- 91 ページの『シナリオのインストール』
- 96 ページの『シナリオの実行』

概要

この単純な例では、eMatrix アダプターは Visual Test Connector と情報を交換します。Visual Test Connector がコンピューター・パーツ調達システムになっているものとします。パーツが利用可能になると、それがコンピューター・メーカーに伝達されます。コンピューター・メーカーは eMatrix アダプターを使用して自動的に eMatrix システムを更新します。コンピューター・メーカーのパーツの要件に変更があると、自動的にその変更が検出され、調達システムに送信されます。

前提事項

前提事項は以下のとおりです。

1. サポートされる統合ブローカーのいずれかをインストール済みで、その運用経験があること (詳細については第 2 章参照)。
2. eMatrix バージョン 9.6.0.1 およびそのコラボレーション・サーバーのいずれかをインストール済みで、その運用経験があること。

シナリオのインストール

最初に、eMatrix をセットアップし、eMatrix ビジネス・オブジェクト・タイプを作成します。

注: %SAMPLE_FOLDER% は、本書が格納されているフォルダーを参照します。%WBIA% は、現行の WebSphere Business Integration Adapters がインストールされているフォルダーを参照します。

eMatrix ビジネス・オブジェクト・タイプの定義

最初に、コンピューター・システムのパーツを表すタイプを定義します。これを行うには、以下のステップを実行します。

1. eMatrix MQL シェルを開きます。
2. 以下のコマンドを、シェルにカット・アンド・ペーストします。

```
#BEGIN
#1. create the attributes

add attribute "component cost"                   type real;
add attribute "in stock"                         type boolean;
add attribute "date available"                   type date;
add attribute "component manufacturer"           type string;
add attribute megahertz                         type integer;
add attribute slot                              type integer;

#2. create the types

add type bundle;
add type computer;
add type cpu attribute "component cost" attribute "in stock" attribute
"date available"
attribute "component manufacturer" attribute megahertz;
add type "network card" attribute "component cost" attribute "in stock"
attribute "date available" attribute "component manufacturer";
add type "usb 2.0 card" attribute "component cost" attribute "in stock"
attribute "date available" attribute "component manufacturer";
add type "sound card" attribute "component cost" attribute "in stock"
attribute "date available" attribute "component manufacturer";
add type "sound chip" attribute "component cost" attribute "in stock"
attribute "date available" attribute "component manufacturer"
attribute megahertz;

#3. create the relationships
add relationship "bundle to computer"' to type computer cardinality n from
type bundle cardinality n;
add relationship "computer to pci component" to type "network card",
"sound card", "usb 2.0 card" cardinality n from type computer
cardinality 1 attribute slot;
add relationship "sound card to sound chip" to type "sound chip"
cardinality 1 from type "sound card" cardinality 1;

#4. create policy

add policy "computer manufacturing" type bundle, computer, cpu,
"network card", "usb 2.0 card", "sound card", "sound chip" state
"pre production" state "ready for production";

#5. create vault

add vault "wbi computer manufacturer";
#6. create the "adapter" person

add person adapter password wbia type business, system access all admin all;

#END
```

上述のステップ 6 では、システム内のすべての操作を実行する権限をユーザー ″adapter″ に与えています。この例を実行するために、これらの特権がすべて必要というわけではありません。以下の MQL コマンドは、必要な特権をすべて提供します。

```
add person adapter password wbia type business access all
admin none;
```

ただし、これらのサンプルで取り上げられていない作業のために eMatrix アダプターを使用する場合、ステップ 6 で提供される特権が必要となることがあります。

eMatrix ビジネス・オブジェクト・タイプの作成

次に、仮定上のコンピューター・メーカーのコンピューター・システムを表すデータを作成します。これを行うには、以下のステップを実行します。

1. eMatrix MQL シェルを開きます。
2. 以下のコマンドを、シェルにカット・アンド・ペーストします。

```
#business objects
add bus bundle "home office and entertainment" 1
    policy                      "computer manufacturing"
    vault                         "wbi computer manufacturer";
add bus computer "gamer deluxe" 1
    policy                      "computer manufacturing"
    vault                         "wbi computer manufacturer";
add bus cpu "max processor" 1
    policy                      "computer manufacturing"
    vault                       "wbi computer manufacturer"
    megahertz                     2000
    "in stock        "            true
    'date available    "     9/30/01
    "component manufacturer" acetech
    "component cost    "     67;
add bus "sound card" "super sonic" 1
    policy                      "computer manufacturing"
    vault                       "wbi computer manufacturer"
    "in stock        "           false
    "date available"     6/12/03
    "component cost"     50
    "component manufacturer" "feedback inc.";
add bus "sound chip" "super sound chip" 1
    policy                      "computer manufacturing"
    vault                       "wbi computer manufacturer"
    "component manufacturer" ChipsWeMake
    megahertz                   100;
add bus "usb 2.0 card" "usb enabler" 1
    policy                      "computer manufacturing"
    vault                       "wbi computer manufacturer"
    "in stock"    '           true
    "date available    "     2/10/02
    "component manufacturer" usbworks
    "component cost"     12;
add bus "network card" "net connector" 1
    policy                      "computer manufacturing"
    vault                       "wbi computer manufacturer"
    "in stock        "            true
    "date available    "     8/12/02
    "component cost"'     10
    "component manufacturer" "tcp specialists";
#relationships
connect bus bundle "home office and entertainment" 1 relationship
"bundle to computer" to computer "gamer deluxe" 1;
connect bus computer "gamer deluxe" 1 relationship "computer to pci component"
to "sound card" "super sonic" 1
slot 1;
connect bus computer "gamer deluxe" 1 relationship "computer to pci component"
to "usb 2.0 card" "usb enable" 1
slot 2;
connect bus "sound card" "super sonic" 1 relationship "sound card to
sound chip" to "sound chip" "super sound chip" 1;
```

JPO のインストール

2 つの JPO をインストールする必要があります。1 つは WebSphere Business Integration Server Express イベントを作成し、イベント・タイプを保存するためのもので、もう 1 つはイベント検出時にトリガーとして動作するものです。

1. eMatrix Business Modeler を開き、プログラムを作成する権限を持つユーザーとしてログインします。
2. 「**オブジェクト**」>「**新規**」>「**プログラム (program)**」を選択します。
3. 「**名前**」というタイトルのテキスト・ボックスに **WBIEventLogger** と入力します。
4. 「**タイプ**」は「**Java**」を選択します。
5. 「**コード**」タブをクリックします。
6. コードを %WBIA%¥connector¥eMatrix¥dependencies¥WBIEventLogger.java ファイルからこのタブのテキスト・ボックスへカット・アンド・ペーストします。ソース・コードで wbiPrefix および vault の値の設定が必要な場合もあります。詳細については、コード中のコメントを調べてください。
7. 「**作成**」をクリックします。

同じステップを実行して、InstallEventTables プログラムを実行します。ただし以下の点が異なります。

- プログラムに InstallEventTables という名前を付けます。
- コードを %WBIA%¥connector¥eMatrix¥dependencies¥InstallEventTables.java ファイルからカット・アンド・ペーストします。

トリガーのインストール

このセクションでは、仮定上のコンピューター設計に変更があったこと検出するトリガーを作成します。'computer to pci component' 関係においてトリガーを作成します。これを行うには、以下のステップを実行します。

1. eMatrix Business Modeler 内で、ウィンドウの左上の角にある双眼鏡をクリックします。
2. 「**関係**」を選択し、「**検索**」をクリックします。
3. 'computer to pci component' ビジネス・オブジェクトをダブルクリックします。
4. 新規ウィンドウで、「**トリガー (Triggers)**」タブを選択し、「**追加**」をクリックします。
5. トリガーの「**作成**」を選択し、「**OK**」をクリックします。
6. 新規ウィンドウで、「**アクション**」テキスト・ボックスに **WBIEventLogger** と入力します。
7. 「**入力**」テキスト・ボックスに次のように入力します。

   ```
   -method recordEvent ${FROMOBJECTID} n=wbi_computer_poll v=Update
   ```

8. 「**OK**」をクリックしてから、「**編集**」をクリックして、トリガーの作成を終了します。

イベント・ビジネス・オブジェクトのインストール

シナリオのインストールの最後のステップは、イベント・ビジネス・オブジェクトのインストールです。これを行うためには、MQL コマンド行から次のコマンドを実行します。

```
exec program InstallEventTables;
```

eMatrix アダプターの構成

最初に、eMatrix Connector を構成します。これを行うには、以下のステップを実行します。

1. Connector Configurator Express を実行します。
2. %SAMPLE_FOLDER%¥EmatrixConnector.cfg を開き、以下のプロパティーを設定します。

   ```
   RepositoryDirectory property = %SAMPLE_FOLDER%¥repos
   PollFrequency property = key
   ```

3. ApplicationUserName および ApplicationPassword プロパティーを、コネクター実行の対象となるユーザーに設定します。
4. eMatrixServer プロパティーを、接続先の eMatrix Server の名前に設定します。
5. HostName プロパティーを、アダプターが通信する eMatrix コラボレーション・サーバーの URL (ポートを含む) に設定します。
6. EventVault プロパティーを、イベントを保管するボールトの名前に設定します。イベントを保管する新規のボールトを作成することができます。これを行うためには、MQL eMatrix クライアントを実行し、ウィンドウ下部のテキスト・ボックスに 'add vault <*vaultname*>' と入力します。この値が WBIEventLogger JPO 内の **vault** ストリングの値と一致することを確認してください。デフォルトでは、どちらの値も 'WBI_Events' となります。
7. KeepRelations プロパティーを 'true' に設定します。
8. %SAMPLE_FOLDER%¥PortConnector.cfg を開き、以下のプロパティーを設定します。

   ```
   RepositoryDirectory = %SAMPLE_FOLDER%¥repos
   ```

ビジネス・オブジェクトのサポート

アダプターがビジネス・オブジェクトを使用するためには、まずビジネス・オブジェクトをサポートする必要があります。使用するビジネス・オブジェクトをサポートするには、以下のステップを実行します。

1. Connector Configurator Express が開いた状態で、「**サポートされているビジネス・オブジェクト**」タブを選択して、以下のビジネス・オブジェクトを追加します。
   - wbi_computer
   - wbi_computer_poll
2. 各ビジネス・オブジェクトの横にある「**サポートされているエージェント (Agent supported)**」ボックスにチェックマークを付けます。
3. ポート・コネクターで上述のステップを繰り返します。

キューの作成

このサンプル・シナリオでは、ご使用のキュー・マネージャー内でいくつかのキューを定義する必要があります。必要なキューを作成するには、コマンド行から

```
RUNMQSC crossworlds.queue.manager
```

と入力し、以下のコマンドを発行します。

```
DEFINE QL("ADMININQUEUE")
DEFINE QL("ADMINOUTQUEUE")
DEFINE QL("DELIVERYQUEUE")
DEFINE QL("FAULTQUEUE")
DEFINE QL("REQUESTQUEUE")
DEFINE QL("RESPONSEQUEUE")
DEFINE QL("EMATRIXCONNECTOR/RESPONSEQUEUE")
DEFINE QL("SYNCHRONOUSREQUESTQUEUE")
DEFINE QL("SYNCHRONOUSRESPONSEQUEUE")
DEFINE QL("PORTCONNECTOR/SYNCHRONOUSREQUESTQUEUE")
DEFINE QL("PORTCONNECTOR/SYNCHRONOUSRESPONSEQUEUE")
```

start_eMatrix.bat または start_eMatrix.sh ファイル (プラットフォームによって異なる) を開き、これらのファイル中で EMADK および EM_LIB 変数を設定して、始動スクリプトを更新します。

シナリオの実行

この例は、要求処理およびイベント通知という 2 つのアクティビティーについてシナリオを提供します。

シナリオ 1: 要求処理

要求処理のシナリオでは、設計で使用されるコンポーネントの 1 つが利用可能になったことを、調達システムが eMatrix システムに通知します。システムは、更新されたシステム・コンポーネントを示すビジネス・オブジェクトを送信します。

Visual Test Connector の始動

1. 「**スタート**」>「**プログラム**」>「**WebSphere Business Integration Adapters**」>「**ツール**」>「**Visual Test Connector**」を選択します。コネクターが始動します。
2. コネクター・ウィンドウで、「**ファイル**」>「**プロファイルを作成/選択**」を選択します。「コネクター・プロファイル」ウィンドウが表示されます。
3. まだプロファイルを作成していなければ、以下のようにして、プロファイル・ウィンドウでポート・コネクターをエミュレートするプロファイルを作成します。
   - 「**ファイル**」>「**新規プロファイル**」を選択します。
   - 「**参照**」をクリックし、%SAMPLE_FOLDER%¥PortConnector.cfg ファイルを選択します。
   - 「コネクター名」テキスト・ボックスに、Port Connector と入力します。「ブローカー・タイプ」ドロップダウン・メニューから、WMQI を選択します。
   - 「**OK**」をクリックします。
4. すでにプロファイルを作成済みの場合は、それを選択して、「**OK**」をクリックします。

5. プロファイルが準備されたら、「**ファイル**」>「**接続**」を選択します。

Adapter for eMatrix の始動

次のコマンドを実行して、Adapter for eMatrix を始動します。

```
%WBIA%¥connectors¥eMatrix¥start_eMatrix.bat eMatrix wmqi
-c%SAMPLE_FOLDER%¥EmatrixConnector.cfg
```

“wmqi” の値を、英数字の任意の組み合わせ (先頭は英字) で置き換えることができます。

eMatrix コンポーネントの始動

eMatrix コラボレーション・サーバーを始動します。eMatrix システムが存在する Oracle のインスタンスが稼働していることを確認します。次に、「MQL コマンド (MQL Command)」ウィンドウを開きます。

eMatrix の現在の状況のチェック

MQL コマンド行で、次のコマンドを入力します。

```
expand bus "computer" "gamer deluxe" 1 select bus
"attribute[in stock]";
```

超音波サウンド・カードは在庫がないことに注意してください。

要求の送信

これを行うには、以下のステップを実行します。

1. Visual Test Connector で、「**編集**」>「**ビジネス・オブジェクトをロード**」を選択します。
2. ファイル %SAMPLE_FOLDER%¥sound_card_update.bo を選択します。
3. ポップアップ・ウィンドウに任意の名前を入力し、「**OK**」をクリックします。
4. Visual Test Connector の左上隅の「**ビジネス・オブジェクトを送信**」をクリックします。アイコンには、黄色の矢印が 1 つ含まれています。
   Adapter for eMatrix を含むウィンドウに、テキスト・メッセージが表示されます。これらのメッセージは、要求の処理中であることを示します。

eMatrix の更新された状況のチェック

MQL コマンド行で、次のように入力します。

```
expand bus "computer" "gamer deluxe" 1 select bus
"attribute[in stock]";
```

ここで超音波サウンド・カードの在庫があることに注意してください。

データベース内の状況のリセット

このシナリオを再度実行するには、超音波サウンド・カードをリセットして、在庫がないようにする必要があります。これを行うには、次の MQL コマンドを実行します。

```
modify bus "sound card" "super sonic" 1 "in stock" false;
```

シナリオ 2: イベント通知

このシナリオでは、コンピューター・メーカーが、コンピューター・システムの 1 つの構成を変更します。この変更がトリガーとなって、調達システムへ通知が転送されます。

1. 前述のシナリオ 1 の、ステップ 1 から 3 までを繰り返します。
2. 以下のステップに従ってください。

コンピューター構成の更新

コンピューター構成が変更されました。コンピューター・メーカーはコンピューターにネットワーク・カードを追加します。MQL コマンド行で、次のコマンドを実行します。

```
connect bus "computer" "gamer deluxe" 1 relationship
"computer to pci component' to "network card" "net

connector" 1 slot 3;
```

イベントのポーリング

Adapter for eMatrix を含むコマンド・ウィンドウに移動します。ウィンドウの下部で、'p' と入力します。これにより、アダプターがポーリングを行います。

イベントの表示

アダプターは、新規のコンピューター構成を記録し、調達システム (この場合は Visual Test Connector) に転送します。

Visual Test Connector の右側のウィンドウ・ペインを調べます。テキスト 'wbi_computer.Update' を含む行が表示されています。これは、WebSphere ビジネス・オブジェクトとして表される新規のコンピューター構成です。ここでダブルクリックすると、その内容を見ることができます。

コンピューター構成のリセット

このシナリオを再度実行するには、eMatrix ビジネス・オブジェクトを切断する必要があります。これを行うには、次のコマンドを実行します。

```
disconnect bus "computer" "gamer deluxe" 1 relationship
"computer to pci component" to "network card"

"net connector" 1;
```

これにより、サンプル・シナリオのインストールおよびテストが完了します。上述のすべてのステップが正常に実行されると、Adapter for eMatrix を使用可能にする適切なサンプルができます。

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 RTP Laboratory
3039 Cornwallis Road
P.O. BOX 12195
Raleigh, NC 27709-2195
U.S.A

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

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

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

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

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

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

プログラミング・インターフェース情報

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

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

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

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

商標

以下は、IBM Corporation の商標です。

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

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

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

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

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

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

Printed in Japan

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