IBM WebSphere Business Integration Adapters

Adapter for XML ユーザーズ・ガイド

バージョン 3.4.x

IBM WebSphere Business Integration Adapters

Adapter for XML ユーザーズ・ガイド

バージョン 3.4.x

お願い

本書および本書で紹介する製品をご使用になる前に、 85 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 Adapter for XML (5724-H07) バージョン 3.4.x に適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： IBM WebSphere Business Integration Adapters
Adapter for XML User Guide
Version 3.4.x

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.7

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

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

目次

本書について

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

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

対象読者

本書は、WebSphere Business Integration システムの一部としてコネクターをインプリメントする WebSphere コンサルタントおよびお客様を対象としています。本書の情報を利用するには、以下の領域についての知識が必要になります。

- コネクター開発
- ビジネス・オブジェクト開発
- HTTP および HTTPS ベースのアプリケーション・アーキテクチャー

本書の前提条件

本書を利用するには、WebSphere Business Integration Adapter システム、ビジネス・オブジェクト開発、およびデータ・ハンドラーについての知識が必要です。また、XML マークアップ言語およびスキーマ言語 (文書タイプ定義 (DTD) または XSDL (スキーマ文書の場合) のどちらか) についての知識も必要です。

関連文書

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

以下のサイトから、関連資料をインストールすることができます。

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

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

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

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

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

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

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

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

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

注: 本書の発行後に公開されたテクニカル・サポートの技術情報や速報に、本書の対象製品に関する重要な情報が記載されている場合があります。これらの情報は、WebSphere Business Integration Support Web サイト (http://www.ibm.com/software/integration/websphere/support) にあります。

関心のあるコンポーネント・エリアを選択し、「Technotes」セクションと「Flashes」セクションを参照してください。

表記上の規則

本書では、以下の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。
青字のテキスト	オンラインで表示したときにのみ見られる青の部分は、相互参照用のハイパーリンクです。青字のテキストをクリックすることにより、参照先オブジェクトにジャンプすることができます。
`{ }`	構文の記述行の場合、中括弧 {} で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
`[ ]`	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプションのパラメーターです。
`...`	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、`option[,...]` は、複数のオプションをコンマで区切って指定できることを示します。
`< >`	命名規則により、1 つの名前の個々の要素を互いに区別するために、不等号括弧によって個々の要素が囲まれます。例えば、`<server_name><connector_name>tmp.log` のように使用します。

/、¥	本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。 UNIX システムの場合には、円記号 (¥) はスラッシュ (/) に置き換えてください。すべての WebSphere Business Integration システム製品のパス名は、ご使用のシステムにおいてこの製品がインストールされているディレクトリーを基準とした相対パスです。
`%text%` および `$text`	パーセント (%) 記号で囲まれたテキストは、Windows の `text` システム変数またはユーザー変数の値を示します。UNIX 環境での同等の表記は `$ text` であり、UNIX の `text` 環境変数の値を示します。
ProductDir	製品がインストールされているディレクトリーを表します。

本リリースの新機能

バージョン 3.4.x

Adapter for XML は、次の項目が更新されました。

- コネクターは、ポーリング時に他の個別のオブジェクトを含むコンテナー・オブジェクトを選択するか、個別のオブジェクトを全て選択するように構成できるようになりました。この機能をサポートするため、コネクター固有プロパティー ResponseTLO が追加されました。
- アダプターは Base-64 認証をサポートするようになりました。このサポートを可能にするため、2 つのコネクター固有プロパティー Login および Password が追加されました。
- HTTP セッションがタイムアウトするまでに、コネクターがサーバーの応答を待機する時間を決定するタイムアウト値が、コネクター固有プロパティー HTTPTimeout を構成することによって設定できるようになりました。

  これらの新しいプロパティーの詳細については、11 ページの『コネクター固有のプロパティー』を参照してください。
- アダプターは IBM JSSE のみサポートするようになりました。
- バージョン 3.4.x から、アダプターは Solaris 7 プラットフォームでサポートされなくなりました。

バージョン 3.3.x

Adapter for XML は一般的な保守フィックスにより更新されました。

バージョン 3.3 以降の Adapter for XML は Microsoft Windows NT ではサポートされなくなりました。

アダプターのインストール情報は、本書から移動しました。この情報の新たな入手先については、10 ページの『XML アダプターのインストール』 (第 2 章) を参照してください。

以前のリリース

以下は、以前のバージョンにおける変更点の履歴です。

バージョン 3.2.x

本ユーザーズ・ガイドに、新規付録として 75 ページの『付録 C. XML Adapter のサンプル・シナリオの概要』が追加されました。この付録には、XML Adapter のテスト手順が記載されています。

アダプターは、WebSphere Application Server を統合ブローカーとして使用できるようになりました。詳細については、9 ページの『互換性』を参照してください。

アダプターは以下のプラットフォーム上で稼働します。

- Solaris 7、8
- AIX 5.x

バージョン 3.1.x

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

バージョン 3.0.x

IBM WebSphere Business Integration Adapter for XML に同梱されるコネクターは国際化されています。詳細については、 7 ページの『ロケール依存データの処理』および 37 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

バージョン 2.5.x

IBM WebSphere Business Integration Adapter for XML には、XML 用のコネクターが含まれます。このアダプターは、InterChange Server (ICS) と WebSphere MQ Integrator の両方の統合ブローカーと共に動作します。統合ブローカーとは、異種のアプリケーション・セット間の統合を実行するアプリケーションです。統合ブローカーは、データ・ルーティングなどのサービスを提供します。

このアダプターには、以下の要素が含まれます。

- XML に固有のアプリケーション・コンポーネント
- サンプル・ビジネス・オブジェクト (格納場所: `¥connectors¥XML¥samples` ディレクトリー)
- IBM WebSphere Adapter フレームワーク。コンポーネントは以下のとおりです。
  - コネクター・フレームワーク
  - 開発ツール (Business Object Designer と Connector Configurator を含む)
  - API (ODK、JCDK、および CDK を含む)

本書では、このアダプターを InterChange Server (ICS) と WebSphere MQ Integrator Broker の両方の統合ブローカーと共に使用するための情報を提供します。

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

第 1 章 XML アダプターの概要

この章では、IBM WebSphere Business Integration Adapter for XML のコネクター・コンポーネントについて説明します。統合ブローカーは、コネクターにより、HTTP および HTTPS プロトコルを使用して URL とのビジネス・オブジェクトの交換を実現できます。URL は、リモート・アプリケーションや Web サーバー上のサーブレットなど、どのような宛先でも差し支えありません。コネクターは XML バージョン 1.0 をサポートしています。

コネクターは、アプリケーション固有のコンポーネントとコネクター・フレームワークから成り立っています。アプリケーション固有のコンポーネントには、特定のアプリケーションに応じて調整されたコードが含まれています。コネクター・フレームワークのコードは、すべてのコネクターに共通です。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントの間を中継します。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントの間で以下のサービスを提供します。

- ビジネス・オブジェクトの受信と送信
- 始動メッセージと管理メッセージの交換の管理

本書では、アプリケーション固有のコンポーネントとコネクター・フレームワークについての情報を提供します。本書では、この 2 つのコンポーネントをまとめてコネクターと呼びます。

統合ブローカーとコネクターの関係の詳細については、「*IBM WebSphere InterChange Server システム管理ガイド*」または「*IBM WebSphere Business Integration WebSphere MQ Integrator Broker 用インプリメンテーション・ガイド*」を参照してください。

注: XML 環境で作業するときには、製品が提供するコネクターを使用するか、カスタム・モジュールを作成するかを選択できます。どちらを使用するかを決定するガイドラインについては、23 ページの『コネクターの実装計画』を参照してください。

この章には、以下のセクションが含まれています。

コネクター・コンポーネント

アダプターは Java で記述され、3 つのコンポーネントから構成されています。

- コネクター
- XML データ・ハンドラー
- プロトコル・ハンドラー (HTTP および HTTPS)

コネクターは XML データ・ハンドラーと対話します。データ・ハンドラーの詳細については、「データ・ハンドラー・ガイド」を参照してください。

図 1 に、コネクター・コンポーネントのアーキテクチャーを示します。コネクターはモジュール構成であるため、製品が提供する機能を置き換えるカスタム・コンポーネントを設計することができます。

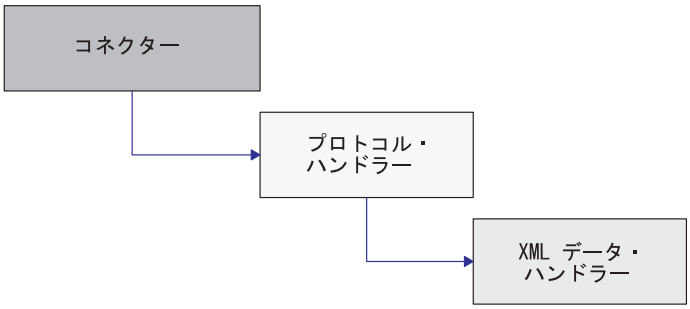

図 1. コネクター・アーキテクチャー

コネクター

コネクターは、統合ブローカーとプロトコル・ハンドラーとの間で、ビジネス・オブジェクトをやりとりします。コネクターは、以下の処理を実行します。

- 統合ブローカーからビジネス・オブジェクト要求を受け取ります。
- プロトコル・ハンドラー・フレームワークを呼び出し、ビジネス・オブジェクトの URL ストリングを渡すことにより、プロトコル・ハンドラーの該当するインスタンスを起動します。
- ビジネス・オブジェクト要求をプロトコル・ハンドラーに渡します。
- プロトコル・ハンドラーから、ビジネス・オブジェクト応答または成功/失敗の戻りコードを受け取ります。コネクターが同期プロトコル・ハンドラーを使用している場合は、ビジネス・オブジェクト応答を受け取ります。コネクターが非同期プロトコル・ハンドラーを使用している場合は、戻りコードに従って成功または失敗を報告します。

コネクターが使用する主なメソッドは、`init()`、`doVerbFor()`、および `pollForEvents()` です。`init()` メソッドは、統合ブローカーのリポジトリーからすべての構成値を読み取り、プロキシー名 (HTTP および HTTPS) とそれに対応するポートを設定し、プロトコル・ハンドラー (`JavaProtocolHandlerPkgs`) および XML データ・ハンドラー (`JavaDataHandlerPkgs`) に対応する Java クラス・パッケージ名、さらにデータ・ハンドラーとプロトコル・ハンドラーのプロパティー値を読み取ります。

`doVerbFor()` メソッドは、ビジネス・オブジェクト要求/応答操作を処理します。コネクターが統合ブローカーからトップレベルのビジネス・オブジェクトを受け取ると、`doVerbFor()` メソッドが要求ビジネス・オブジェクトと宛先 URL を抽出します。次に、`doVerbFor()` メソッドが適切なプロトコル・ハンドラー・インスタンスを作成します。

コネクターが宛先 URL から応答を受け取ると、doVerbFor() メソッドはトップレベルのビジネス・オブジェクトの子としての応答ビジネス・オブジェクトにデータを設定し、その結果を統合ブローカーに戻します。コネクターの中で、すべてのエラーは例外として伝搬し、コネクターにより処理されます。その結果、BON_FAIL が戻り、Return Status Descriptor が設定されます。

pollForEvents() メソッドは、イベント通知用に使用されます。コネクターには、ビジネス・オブジェクトを使用して URL からのイベントをチェックする機能があります。イベント通知の詳細については、7 ページの『イベント通知』 を参照してください。

コネクターは、データ・ハンドラーのトップレベルのメタオブジェクト名を、DataHandlerConfigMO コネクター構成プロパティーで指定されたとおりに静的プロパティーに設定します。

プロトコル・ハンドラー (HTTP および HTTPS)

コネクターは、プロトコル・ハンドラーにより、HTTP および HTTPS プロトコルを使用して URL との通信を実現できます。プロトコル・ハンドラーは、Java URLConnection クラスから拡張された抽象基本クラスです。このクラスに所属する抽象メソッドを実装することにより、HTTP や HTTPS など具体的なプロトコルをサポートすることができるようになります。コネクターから呼び出されるプロトコル・ハンドラー・フレームワークにより、プロトコル・ハンドラーのインスタンスが作成されます。

WebSphere Business Integration Adapter for XML には同期と非同期、両方のプロトコル・ハンドラーが含まれています。同期プロトコル・ハンドラーは、同期応答に基づいてビジネス・オブジェクトを戻します。非同期プロトコル・ハンドラーは応答ビジネス・オブジェクトの受信を想定していません。通知動作の結果として受け取る戻りコードに基づいて成功または失敗のメッセージを戻します。非同期プロトコル・ハンドラーはイベント通知をサポートしていません。

注: プロトコル・ハンドラー・フレームワークを使用することにより、FTP など他のプロトコルのサポートを追加することができます。プロトコル・ハンドラー・フレームワークは、CWURLConnection と呼ばれる抽象基本クラスです。

プロトコル・ハンドラー・フレームワークは、プロトコル・ハンドラーのインスタンスを作成し、この作成されたインスタンスに、コネクターがビジネス・オブジェクトを渡します。プロトコル・ハンドラーは、ビジネス・オブジェクトから内容タイプ (text/plain や text/xml など) を抽出し、これを使用して XML データ・ハンドラーのインスタンスを作成します。

プロトコル・ハンドラーが createHandler() メソッドを呼び出すと、メソッドは内容タイプで渡されます。データ・ハンドラー作成メソッドは、スラッシュ (/) をピリオド (.) に、すべての非英数字文字を下線 (_) に置き換えることにより、内容タイプを渡します。次に作成メソッドは、内容タイプから解析されたストリングに一致する属性を、データ・ハンドラーのトップレベルのメタオブジェクトから探します。一致が見つからない場合、メソッドはクラス名を com.crossworlds.DataHandlers.*modified content_type* としてクラスを構築します。

プロトコル・ハンドラーは次の処理を実行します。

- コネクターからビジネス・オブジェクトを受け取り、XML データ・ハンドラーに渡します。プロトコル・ハンドラーは MimeType 属性を解析することにより、どのデータ・ハンドラー・インスタンスを作成するか決定します。
- XML データ・ハンドラーから XML ストリームを受け取り、適切な URL にこれを渡します。XML ストリームは、要求ビジネス・オブジェクトを表します。

  データ・ハンドラーが XML ストリングを解析すると、プロトコル・ハンドラーは、この XML ストリング を XML ストリームに変換してから、URL に渡します。
- プロトコル・ハンドラーが同期式の場合、URL から応答ストリームを受け取り、それを XML データ・ハンドラーに戻します。データ・ハンドラーは受け取った応答を変換して WebSphere Business Integration Adapter ビジネス・オブジェクトに戻します。
- プロトコル・ハンドラーが非同期式の場合、要求処理からの戻りコードに基づいて成功または失敗を URL に報告します。
- 応答ビジネス・オブジェクトをコネクターに戻します。

実装されたコネクターに追加プロトコルのサポートが必要な場合には、カスタム・プロトコル・ハンドラーの作成が必要です。カスタム・プロトコル・ハンドラーの作成方法については、31 ページの『第 4 章 カスタム・プロトコル・ハンドラーの作成』を参照してください。

コネクターの動作方法

以降のセクションでは、コネクターがビジネス・オブジェクト要求を処理する方法、構成のためにメタオブジェクトを使用する方法、およびコネクターがイベント通知を処理する方法について説明します。

ビジネス・オブジェクトの処理

コネクターは、要求/応答動作により、コネクターと URL の間でデータをやり取りします。コネクターは統合ブローカーからビジネス・オブジェクト要求を受け取り、この要求を XML ストリームに変換します。要求ストリームは、POST メソッドにより URL に渡されます。戻される応答ストリームは、内容が同様な場合と異なる場合とがあります。応答ストリームは応答ビジネス・オブジェクトに変換され、当初のトップレベルのビジネス・オブジェクトと共に統合ブローカーに戻されます。ビジネス・オブジェクト要求のタイプは、ビジネス・オブジェクト応答のタイプと異なる場合もあるので注意してください。

図 2 に、要求/応答サイクルの全体を示します。

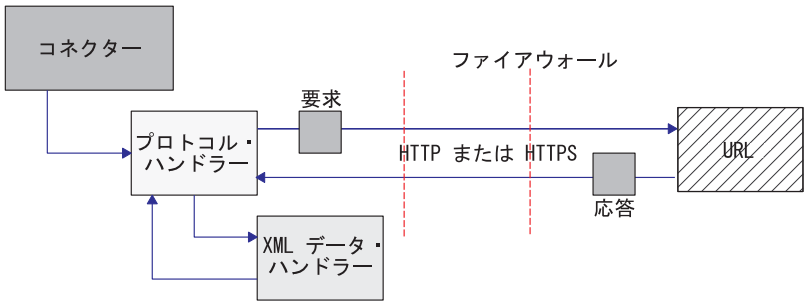

図 2. ビジネス・オブジェクトのイベント処理

要求

統合ブローカーからビジネス・オブジェクト要求を受け取ったコネクターは、この要求を、適切なプロトコルを使用して渡すことのできる要求ストリームに変換する必要があります。要求ビジネス・オブジェクトの変換と URL への送信には、プロトコル・ハンドラーと XML データ・ハンドラーが使用されます。図 3 に要求プロセスを示します。

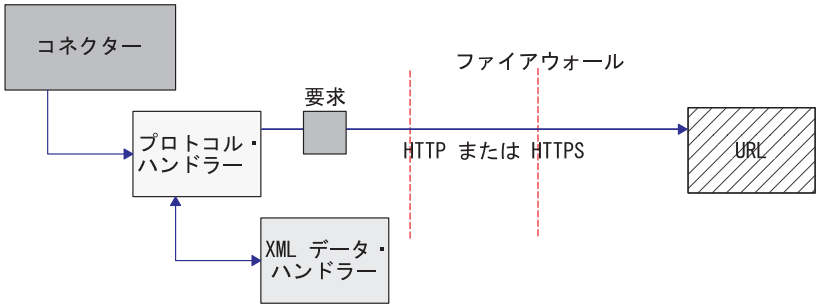

図 3. 要求処理

特に、コネクターがトップレベル・ビジネス・オブジェクトを統合ブローカーから受け取った場合の処理フローは次のようになります。

1. コネクターは、`getAttrValue ("URL")` を呼び出し、URL を検索します。また、`getAttrValue ("MimeType")` と `getAttrValue ("BOPrefix")` を呼び出すことにより、ビジネス・オブジェクトから MimeType と BOPrefix の属性値を検索します。
2. コネクターは、トップレベルのビジネス・オブジェクトから要求ビジネス・オブジェクトを抽出します。
3. コネクターは、トップレベルのビジネス・オブジェクトの URL フィールドに指定されたプロトコルおよび指定されたプロトコル・ハンドラー・パッケージ名に基づいて、適切なプロトコル・ハンドラー (HTTP または HTTPS) を呼び出します。
4. プロトコル・ハンドラーは、トップレベルのビジネス・オブジェクト (トップレベルのメタオブジェクトの中に構成されたもの) の MimeType および BOPrefix 属性に基づいて、適切なデータ・ハンドラーを呼び出します。

5. データ・ハンドラーは、ビジネス・オブジェクトを要求ストリームに変換し、これを、プロトコル・ハンドラーに戻します。
6. プロトコル・ハンドラーは、トップレベルのビジネス・オブジェクトに指定された宛先 URL に要求ストリームを送るか、あるいは戻りコードを渡します。

応答

同期式プロトコル・ハンドラーを使用している場合、応答ビジネス・オブジェクトが URL から戻るとき、応答ストリームの形式で戻ります。非同期式プロトコル・ハンドラーを使用している場合には、戻りコードがそのまま戻ります。応答処理は要求処理と同様ですが、応答ストリームをビジネス・オブジェクトに変換して戻す必要のある点が違います。

注: 応答ストリームは、要求ストリームと同じビジネス・オブジェクト・タイプで表現されない場合もあります。

図 4 に、応答ビジネス・オブジェクトがコネクターに戻る処理フローを示します。

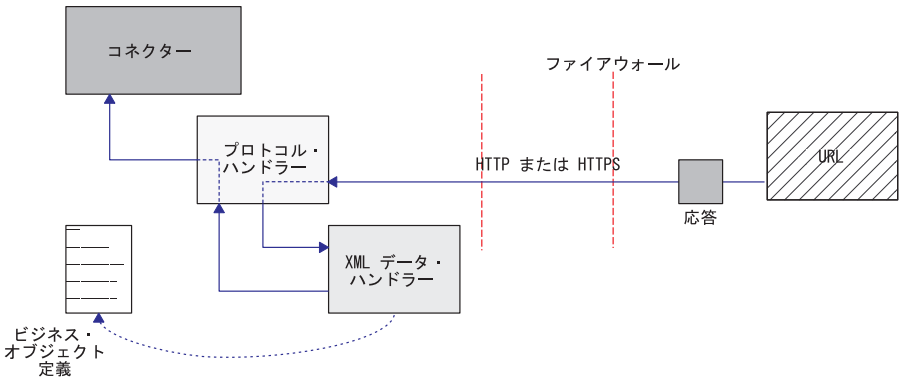

図 4. URL からデータが戻るときの処理フロー

特に、プロトコル・ハンドラーが URL から応答ストリームを受け取ったとき、MIME タイプが `text/xml` の場合の処理フローは次のようになります。

1. プロトコル・ハンドラーは `getContentType ()` メソッドを呼び出すことにより、MIME タイプの値を取得し、どのデータ・ハンドラーを使用するか決定します。
2. プロトコル・ハンドラーは、`DataHandler` クラスを呼び出すことにより、XML データ・ハンドラーのインスタンスを作成します。

   応答ストリームのデータ形式は、当初の要求ビジネス・オブジェクト内のデータ形式と異なる場合があります。
3. プロトコル・ハンドラーは応答ストリームをストリングに変換し、このストリングを XML データ・ハンドラーに渡します。
4. XML データ・ハンドラーは、メッセージの内容に基づいてビジネス・オブジェクト名を取得し、応答ストリーム (XML 文書) からデータを抽出し、このデータからビジネス・オブジェクトを作成します。
5. XML データ・ハンドラーは完成した応答ビジネス・オブジェクトをプロトコル・ハンドラーに渡します。

6. プロトコル・ハンドラーは、応答ビジネス・オブジェクトをコネクター・エージェントに渡し、コネクターは、受け取ったビジネス・オブジェクトを当初のトップレベルのビジネス・オブジェクトに追加します。
7. コネクターは、応答ビジネス・オブジェクトが格納された当初のトップレベルのビジネス・オブジェクトを、統合ブローカーに戻します。

イベント通知

イベント通知では、コネクターは、ビジネス・オブジェクトを使用して、URL からイベントを取り込みます。コネクターは、応答 XML 文書として戻った要求 XML 文書を送ることにより、URL のポーリングを実行します。応答には、コネクターからイベントとして統合ブローカーに渡された子ビジネス・オブジェクトが格納されています。各子ビジネス・オブジェクトは 1 つのイベントとして処理されます。非同期プロトコル・ハンドラーはイベント通知をサポートしていません。

注: イベント処理のためのポーリングは、ビジネス・オブジェクト要求処理と同じですが、応答ビジネス・オブジェクトからイベント・オブジェクトを抽出し、統合ブローカーに送るステップが追加されている点のみが異なっています。

イベント通知用のビジネス・オブジェクトは、XML ビジネス・オブジェクトの要求および応答ビジネス・オブジェクトと同じビジネス・オブジェクト処理操作に従います。アンサブスクライブされたイベントはすべてファイルにアーカイブされます。このアーカイブの形式は、WebSphere Business Integration Adapter の標準ビジネス・オブジェクト・ダンプ形式です。

イベント通知を有効にするには、イベント通知ビジネス・オブジェクトを定義し、これらのビジネス・オブジェクトの処理のために URL (Web サーブレットや `cgi-bin` スクリプトなど) を設定することが必要です。コネクターは、POST メソッドを使用して、XML イベント要求文書をストリームとして URL に送ります。URL は XML 文書を STDIN からのストリームとして読み取り、1 つ以上のイベント・オブジェクトを格納した XML 文書をストリームとして STDOUT に書き込みます。

図 5 に、イベント通知の基本プロセスを示します。

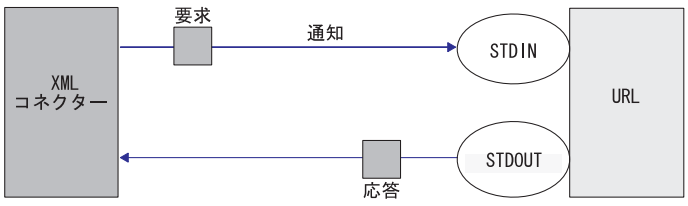

図 5. イベント通知プロセス

ビジネス・オブジェクトの定義の詳細については、23 ページの『第 3 章 コネクター用ビジネス・オブジェクトの開発』を参照してください。

ロケール依存データの処理

コネクターは、2 バイト文字セットをサポートし、指定の言語でメッセージ・テキストを配信できるように国際化されています。コネクターは、1 文字のコードを使

用する地域から異なるコード・セットを使用する地域にデータを転送する場合、文字変換を実行し、データの意味を維持します。Java 仮想マシン (JVM: Java Virtual Machine) 内の Java ランタイム環境では、データが Unicode 文字コード・セットで表現されます。Unicode には、既知の文字コード・セット (単一バイトとマルチバイトの両方) における文字のエンコードが、ほぼ一通り組み込まれています。
WebSphere Business Integration システム内の大部分のコンポーネントは Java で作成されています。したがって、統合コンポーネント間のデータ転送では、大半の場合、文字変換の必要はありません。エラーおよび通知メッセージを、該当する言語で、該当する国または地域に対応してログに記録するには、実際の環境に応じた `Locale` 標準構成プロパティーを構成します。構成プロパティーの詳細については、37 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

第 2 章 コネクターのインストールと構成

この章では、コネクターのインストールと構成のプロセスについて説明します。本章の内容は、次のとおりです。

互換性

アダプターが使用するアダプター・フレームワークは、アダプターと通信する統合ブローカーのバージョンとの互換性を備えている必要があります。Adapter for XML アダプターのバージョン 3.4 は、以下の統合ブローカーを使用する以下のバージョンのアダプター・フレームワークでサポートされます。

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

統合ブローカー:

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

例外については、「リリース情報」を参照してください。

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

Message Brokers (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) の場合は、「WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド」およびそ

れぞれの Message Broker のインストールに関する資料を参照してください。一部の資料は次の Web サイトにあります。
http://www.ibm.com/software/integration/mqfamily/library/manualsa/

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

前提条件

コネクターを使用するための環境条件は次のとおりです。

- JDK 1.2 以降
- Java Secure Socket Extension 1.0 (JSSE)

  WebSphere Business Integration Adapter は国際バージョンを提供します。国内グレード暗号化されたバージョンをダウンロードして、
  `¥connector¥Xml¥dependencies` ディレクトリーに追加することも可能です。
- 宛先 URL へのアクセス

XML アダプターのインストール

WebSphere Business Integration アダプター製品のインストールについては、次のサイトで WebSphere Business Integration Adapters Infocenter にある「*WebSphere Business Integration Adapters インストール・ガイド*」を参照してください。

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

コネクターの構成

WebSphere MQ Integrator Broker を統合ブローカーとして使用している場合は、Connector Configurator からコネクター・プロパティーを構成します。WebSphere ICS を統合ブローカーとして使用している場合は、Connector Configurator からコネクター・プロパティーを構成します。Connector Configurator には System Manager からアクセスできます。

データ・ハンドラーの構成

XML データ・ハンドラー用に使用されるメタオブジェクトを構成します。メタオブジェクトの構成については、15 ページの『データ・ハンドラー用のトップレベルのメタオブジェクトの構成』を参照してください。

標準コネクター・プロパティー

標準構成プロパティーは、すべてのコネクターが使用する情報を提供します。これらのプロパティーの詳細については、37 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

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

表 1 は、この付録に含まれる構成プロパティーのうち、このコネクターに固有な構成プロパティーの情報を示したものです。

表 1. このコネクターに固有のプロパティー情報

プロパティー	注
CharacterEncoding	このコネクターはこのプロパティーを使用しません。
Locale	このコネクターは国際化されているため、このプロパティーの値は変更可能です。

コネクターを実行するには、ApplicationName 構成プロパティーの値を指定する必要があります。また、コネクターを実行するには、少なくとも次の標準コネクター構成プロパティーを設定する必要があります。

- AgentTraceLevel
- ApplicationName
- ControllerStoreAndForwardMode
- ControllerTraceLevel
- DeliveryTransport

コネクター固有のプロパティー

コネクター固有の構成プロパティーは、コネクターが実行時に必要とする情報を提供します。コネクター固有のプロパティーは、コネクターを再コーディングまたは再ビルドせずに、コネクター内部の静的情報またはロジックを変更する手段にもなっています。

表 2 に、コネクターのコネクター固有構成プロパティーのリストを示します。プロパティーの説明については、次のセクションを参照してください。

表 2. コネクター固有のプロパティー

名前	指定可能な値	デフォルト値	必要
ArchiveDirectory	アーカイブ・ディレクトリー名	¥connectors¥xml¥ archive	
DataHandlerConfigMO	データ・ハンドラーのメタオブジェクト名	MO_DataHandler_ Default	はい
HttpProxyHost	*HTTP* ホスト名		
HttpProxyPort	*HTTP* プロキシー・ポート	80	
HttpTimeout	HTTP セッションがタイムアウトになるまでコネクターが待機する時間。	0	
HttpsDebug	プロパティーを 13 ページの表 3 内のいずれかの値に設定します。		
HttpsProxyHost	HTTPS ホスト名		
HttpsProxyPort	HTTPS プロキシー・ポート	443	
JavaProtocolHandlerPkgs	プロトコル・ハンドラー名	com.crossworlds. connectors.utils. ProtocolHandlers	はい
Login	ユーザー・ログイン		
MaxNumRetries	正整数	10	
Password	ユーザー・パスワード		
PollingBusinessObjects	ビジネス・オブジェクト名		
ReturnBusObjResponse	true または false	true	
ResponseTLO	true または false	false	
SecurityProvider	実装 SSL	com.ibm.ssl.jsse.JSSEProvider	はい
UseCaches	true または false	false	
UseDefaults	true または false	false	
UseDigitalSignature	true または false	false	

ArchiveDirectory

イベントのアーカイブを格納するディレクトリーです。各イベントは、そのビジネス・オブジェクト名および動詞により識別できます。デフォルトでは、ビジネス・オブジェクト名の後ろに Create 動詞が付きます。デフォルトは ¥connectors¥xml¥ archive です。

DataHandlerConfigMO

XML コネクターがそのデータ・ハンドラー・サポートを決定するために使用するトップレベルのメタオブジェクトの名前です。このメタオブジェクトは、XML データ・ハンドラーが構成プロパティーを設定するために使用する子メタオブジェクトの名前を格納していなければなりません。このプロパティーは、特定の内容タイプのためにインスタンス化する DataHandler クラスを決定するために、DataHandler 基本クラスも使用します。デフォルトは MO_DataHandler_Default です。詳細については、15 ページの『データ・ハンドラー用のトップレベルのメタオブジェクトの構成』を参照してください。

HttpProxyHost

HTTP 用のプロキシーとして機能するホストの名前です。このプロパティーは、HTTP プロトコルを使用するプロキシー・サーバーがネットワークで運用されている場合にのみ必要です。

HttpProxyPort

HTTP の接続に使用するプロキシーのポート番号です。このプロパティーは、HTTP プロトコルを使用するプロキシー・サーバーがネットワークで運用されている場合にのみ必要です。デフォルトのポート番号は 80 です。

HttpTimeout

HTTP セッションがタイムアウトになるまでに、サーバーからの応答をコネクターが待機する時間 (ミリ秒)。

デフォルト値は 0 に設定されています。

HttpsDebug

HTTPS セッション用に生成されるデバッグ情報を決定する設定です。表 3 に、HTTPS プロトコル・ハンドラーの HTTPS デバッグ値を示します。

表 3. HttpsDebug 値

名前	意味
all	すべてのデバッグをオンにします。
data	各ハンドシェーク・メッセージの 16 進ダンプです。この設定は、ハンドシェーク・デバッグの拡張に使用できます。
handshake	各ハンドシェーク・メッセージを印刷します。この設定は SSL で使用できます。
keygen	キー生成データを印刷します。この設定は SSL で使用できます。
plaintext	レコードのプレーン・テキストの 16 進ダンプです。この設定は、レコード・デバッグの拡張に使用できます。
record	レコード単位のトレースを有効にします。この設定は SSL で使用できます。
session	セッション・アクティビティーを印刷します。この設定は SSL で使用できます。
ssl	SSL デバッグのみをオンにします。
verbose	冗長ハンドシェーク・メッセージを印刷します。この設定は、レコード・デバッグの拡張に使用できます。

HttpsProxyHost

HTTPS プロキシー・マシンの名前です。このプロパティーは、HTTPS プロトコルを使用するプロキシー・サーバーがネットワークで運用されている場合にのみ必要です。

HttpsProxyPort

HTTPS の接続に使用するプロキシーのポート番号です。このプロパティーは、HTTPS プロトコルを使用するプロキシー・サーバーがネットワークで運用されている場合にのみ必要です。

JavaProtocolHandlerPkgs

この属性が存在していると、デフォルトの Java ハンドラーではないプロトコル・ハンドラーとして使用されるパッケージを指定します。これらのクラスは、Java のプロトコル・ハンドラー・フレームワークに準拠していることが必要です。例えば、com.mycompany.http (HTTP 用) という名前のプロトコル・ハンドラーを使用するに

は、このフィールドを com.mycompany に設定します。また、対応するクラスの .jar ファイルがクラスパス内にあることを確認してください。

Java プロトコル・ハンドラーの詳細については、次の Web サイトにあるチュートリアルを参照してください。
http://developer.java.sun.com/developer/onlineTraining/protocolhandlers/

com.crossworlds.Protocol Handlers|com.mycompany のように、この値に対して、複数のパッケージを縦線 (|) で区切って指定することもできます。

WebSphere Business Integration Adapter は次の 2 つのパッケージを提供しています。

- com.crossworlds.connectors.utils.ProtocolHandlers (同期プロトコル・ハンドラー)
- com.crossworlds.connectors.utils.ProtocolHandlers.async (非同期プロトコル・ハンドラー)

デフォルトは、com.crossworlds.connectors.utils.ProtocolHandlers です。

Login

ユーザーのログイン ID です。Base-64 認証をサポートする認証クリデンシャルとして渡されます。

MaxNumRetries

非同期プロトコル・ハンドラーが相手 URL から応答を受信しなかったときの再試行回数を指定します。このプロパティーは、非同期プロトコル・ハンドラーのみが使用します。値を指定しなければ、このプロパティーは 0 と解釈されます。デフォルトは 0 です。

Password

ユーザーのパスワードです。Base-64 認証をサポートする認証クリデンシャルとして渡されます。

PollingBusinessObjects

イベント通知に使用されるビジネス・オブジェクトです。複数の項目をコンマで区切ります (XMLPoll_Cust、XMLPoll_Order など)。各ビジネス・オブジェクトはコネクターによりサポートされていることが必要です。このプロパティーは、コネクターがイベント通知用に設定されている場合に必要です。

ReturnBusObjResponse

コネクターが、プロトコル・ハンドラーからビジネス・オブジェクトが戻ることを予測するかどうかを指定します。この値を true に設定すると、コネクターはビジネス・オブジェクトの戻りを予測します。この値を false に設定すると、コネクターはビジネス・オブジェクトの戻りを予測しません。成功または失敗の応答のみを予測します。デフォルトは true です。

注: 非同期プロトコル・ハンドラーを構成している場合には、ビジネス・オブジェクトの応答は予測されないため、この値は false に設定することが必要です。

ResponseTLO

ポーリング時に使用され、アダプターが、他のオブジェクトを含むコンテナー・オブジェクトを選択するか、個々のオブジェクトを選択するかを指定します。true に設定すると、アダプターはコンテナー・オブジェクトを渡します。false に設定すると、アダプターは個々のオブジェクトを渡します。

デフォルト設定は false です。

SecurityProvider

SSL ハンドシェークのとき、HTTPS により使用されます。この属性に指定されたコンマで区切った値により、HTTPS URL に接続するときどの実装の SSL を使用するかが決まります。値の設定がなければ、HTTPS 接続は機能しません。デフォルトは com.ibm.jsse.JSSEProvider です。

UseCaches

この属性が false に設定されていると、コネクターはキャッシュにないバージョンの XML 文書を取得しようとします。これは単なる要求であり、コネクターから厳格に強制することはできません。キャッシュされた XML 文書のみを検索するには、この値を true に設定します。

UseDefaults

Create 操作では、UseDefaults を true に設定した場合に、各 isRequired ビジネス・オブジェクト属性に対して有効な値またはデフォルト値が指定されているかどうかをコネクターがチェックします。値が指定されている場合は Create は正常に実行されます。パラメーターが false に設定されていると、コネクターは有効な値のみチェックし、値が設定されていなければ、Create 処理は失敗します。デフォルトは false です。

UseDigitalSignature

HTTP または HTTPS プロトコルを使用して送信されるメッセージの最後に、ディジタル・シグニチャー長 (2 進数の 0) を追加するかどうかを指定します。コネクターがディジタル・シグニチャーをサポートしている場合には、このプロパティーを true に設定します。デフォルトは false です。

注: 製品が提供するコネクターは、ディジタル・シグニチャーをサポートしていません。

データ・ハンドラー用のトップレベルのメタオブジェクトの構成

メタオブジェクトは、構成情報を格納しているビジネス・オブジェクトです。データ・ハンドラーのメタオブジェクトには、データ・ハンドラーを構成するために情報が格納されています。コネクターは、データ・ハンドラー・メタオブジェクト内の情報を使用して、XML データ・ハンドラーのインスタンスを作成します。

コネクターを起動する前に、データ・ハンドラー・メタオブジェクトを設定し、これにより、コネクターが MIME タイプに基づいてどのデータ・ハンドラーを使用するかを指定する必要があります。コネクターは、ビジネス・オブジェクト要求を受け取ると、メタオブジェクト内の情報を使用して、適切なデータ・ハンドラーのインスタンスを動的に作成します。

コネクターは、コネクター構成プロパティー DataHandlerConfigMO からトップレベルのデータ・ハンドラー・メタオブジェクトの名前を取得します。トップレベルのメタオブジェクトは、階層構造をもつビジネス・オブジェクトであり、ここに複数の子オブジェクトを格納することができます。各子オブジェクトは、特定のデータ・ハンドラー・インスタンスを表すフラット (階層構造を持たない) オブジェクトです。子メタオブジェクトには属性があり、属性に指定された構成値により、データ・ハンドラー・インスタンスはその機能を実現することができます。データ・ハンドラーのタイプが異なれば、必要な構成プロパティーも異なります。したがって、特定のハンドラーをサポートしている子メタオブジェクトも属性がそれぞれ異なります。

XML コネクターは、XML データ・ハンドラーを使用することにより、ビジネス・オブジェクトと XML 文書間の変換を実行します。コネクターのために XML データ・ハンドラーを構成する手順は次のとおりです。

- トップレベルのデータ・ハンドラー・メタオブジェクトが、コネクターがサポートする各 MIME タイプに対して属性を持つように設定します。属性名は MIME タイプの名前にしてください。属性は、データ・ハンドラー・インスタンスに対応する子メタオブジェクトを表します。

  XML コネクターの場合は、`text/xml` MIME タイプに対応する属性がトップレベルのメタオブジェクトに格納されていることを確認します。この属性には、XML データ・ハンドラーに対応する子メタオブジェクトの名前も設定されていることが必要です。
- それぞれの子メタオブジェクトにデフォルトの属性値を設定します。WebSphere Business Integration Adapter データ・ハンドラーの構成プロパティーについての説明は、「*データ・ハンドラー・ガイド*」にあります。

  XML データ・ハンドラーに対応する子メタオブジェクトに、適切なデフォルト属性値を設定します。

個別データ・ハンドラーに対応するメタオブジェクトの設定の詳細については、「*データ・ハンドラー・ガイド*」を参照してください。

注: コネクターがデータ・ハンドラーのインスタンスを作成するためには、データ・ハンドラーのトップレベルのメタオブジェクトが、コネクターにサポートされているオブジェクトのリストに所属していることが必要です。

一般的な構成作業

このセクションでは通常実行されるコネクターの構成作業について説明します。

イベント通知の設定

コネクターのイベント通知機能を使用可能にするためのステップは次のとおりです。

1. 子要求オブジェクトと応答ビジネス・オブジェクトを格納しているトップレベルのビジネス・オブジェクトを作成します。

2. 要求と応答のビジネス・オブジェクトの構造を処理できるように、URL を構成します。ビジネス・オブジェクトの定義の詳細については、23 ページの『第 3 章 コネクター用ビジネス・オブジェクトの開発』を参照してください。
3. イベント通知ビジネス・オブジェクトを定義した後、WebSphere MQ Integrator Broker が統合ブローカーである場合は、Connector Configurator を使用して PollingBusinessObjects および ArchiveDirectory 構成プロパティーを設定します。InterChange Server が統合ブローカーである場合は、Connector Configurator を使用します。Connector Configurator には System Manager からアクセスします。

データ・ハンドラーの指定

XML コネクターが使用するデータ・ハンドラーを指定するためのステップは次のとおりです。

1. コネクターがサポートするデータ形式のタイプを決定します。

   デフォルトでは、コネクターは `text.xml` MIME タイプ用の XML データ・ハンドラーを使用します。ビジネス・オブジェクトと他の MIME タイプの間で変換を実行する場合には、その MIME タイプがトップレベルのデータ・ハンドラー・メタオブジェクト (デフォルトでは `MO_DataHandler_Default`) 内の属性であることを確認します。1 つの形式タイプの変換に使用できるデータ・ハンドラーは 1 つのみです。
2. コネクターがどのデータ・ハンドラー (1 つまたは複数) を使用するか決定します。

   トップレベルのデータ・ハンドラー・メタオブジェクトにより、MIME タイプと子データ・ハンドラー・メタオブジェクトの対応が付けられます。子データ・ハンドラー・メタオブジェクトにより、どのデータ・ハンドラーのインスタンスが作成されるか決定されます。
3. Business Object Designer を使用して、データ・ハンドラー・メタオブジェクトを変更します。

   注: InterChange Server を統合ブローカーとして使用している場合は、System Manager 内から Business Object Designer を起動することができます。
4. Connector Configurator または System Manager で、コネクターがサポートするオブジェクトのリストに、データ・ハンドラーのトップレベルのメタオブジェクトを追加します。コネクターがトップレベルのデータ・ハンドラー・メタオブジェクトにアンサブスクライブされた場合には、コネクターは始動時にメタオブジェクトをロードしません。
5. コネクターは、コネクターの `DataHandlerConfigMO` 構成プロパティーにトップレベルのデータ・ハンドラー・メタオブジェクトの名前を指定します。製品が提供するデフォルトは `MO_DataHandler_Default` メタオブジェクトです。

データ・ハンドラー・メタオブジェクトの詳細については、「*データ・ハンドラー・ガイド*」を参照してください。

複数のコネクター・インスタンスの作成

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

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

新規ディレクトリーの作成

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

```
ProductDir¥connectors¥connectorInstance
```

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

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

```
ProductDir¥repository¥connectorInstance
```

ビジネス・オブジェクト定義の作成

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

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

   ```
   ProductDir¥repository¥initialConnectorInstance
   ```

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

コネクター定義の作成

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

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

3. 必要に応じて、コネクター・プロパティーをカスタマイズします。

始動スクリプトの作成

始動スクリプトは以下のように作成します。

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

   dirname

2. この始動スクリプトを、18 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. 始動スクリプトのショートカットを作成します (Windows のみ)。
4. 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。

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

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

コネクターの始動

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

ProductDir¥connectors¥*connName*

ここで、*connName* はコネクターを示します。始動スクリプトの名前は、表 4 に示すように、オペレーティング・システム・プラットフォームによって異なります。

表 4. コネクターの始動スクリプト

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

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

- Windows システムの**「スタート」**メニューから。

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

    start_*connName connName brokerName* [-c*configFile*]

– UNIX ベースのシステム:

```
connector_manager_connName -start
```

ここで、*connName* はコネクターの名前であり、*brokerName* は以下のようにご使用の統合ブローカーを表します。

– WebSphere InterChange Server の場合は、*brokerName* に ICS インスタンスの名前を指定します。

– WebSphere Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、または WebSphere Business Integration Message Broker) または WebSphere Application Server の場合は、*brokerName* にブローカーを示すストリングを指定します。

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

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

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

  このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。
- Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムがブートしたとき (自動サービスの場合)、または Windows サービス・ウィンドウを通じてサービスを始動したとき (手動サービスの場合) に、コネクターが始動します。

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

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

コネクターの停止

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

- コマンド行からコネクターを始動した場合は、コネクター始動スクリプトを用いて、以下の操作を実行します。

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

```
connector_manager_connName -stop
```

ここで、*connName* はコネクターの名前です。

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

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

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

第 3 章 コネクター用ビジネス・オブジェクトの開発

この章では、コネクターが処理するトップレベルのビジネス・オブジェクトの構造について説明します。また、必須属性について説明するとともに、コネクターがトップレベルのビジネス・オブジェクトを処理する方法についても説明します。次のセクションが含まれています。

- 『コネクターの実装計画』
- 24 ページの『コネクター・ビジネス・オブジェクトの構造』
- 28 ページの『イベント通知用のビジネス・オブジェクト』
- 29 ページの『XML DTD またはスキーマ文書に基づくビジネス・オブジェクト』

コネクターの実装計画

コネクターはモジュール方式の設計であるため、コネクター全体の整合性を損なうことなく、コンポーネントの交換や追加が可能です。コネクターとそのコンポーネントの構成を開始する前に、開発を必要とするシステムの分析を十分に行ってください。

以下に、コネクターのコンポーネントを、製品で提供されたまま変更なしに使用できるかどうか判断するための情報を提供します。コネクターのあるコンポーネントの機能がニーズに合わない場合には、カスタム・コンポーネントで置き換えることができます。例えば、アプリケーションで XML 以外のデータ型に対処することが予測される場合には、カスタム・データ・ハンドラーの実装が必要となる可能性があります。

表 5 を使用して、提供されたコネクター・コンポーネントをそのまま使用できるか、カスタム・コンポーネントの作成が必要か判断してください。

表 5. WebSphere Business Integration Adapter コンポーネントの使用あるいはカスタム・コンポーネントの作成

WebSphere Business Integration Adapter 提供のコンポーネント	**左記コンポーネントを使用できる条件 (すべてが満たされること)**	**作成が必要なカスタム・コンポーネント**
同期プロトコル・ハンドラー (HTTP/HTTPS)	• HTTP または HTTPS プロトコルを使用 • ユーザー/パスワードの交換が使用可能 • 認証用の詳細情報が不要 • URL からの応答ビジネス・オブジェクトが必要	カスタム・プロトコル・ハンドラー (*31 ページの『第 4 章 カスタム・プロトコル・ハンドラーの作成』*を参照)
非同期プロトコル・ハンドラー (HTTP/HTTPS)	• HTTP または HTTPS プロトコルを使用 • ユーザー/パスワードの交換が設定可能 • 認証用の詳細情報が不要 • URL から成功または失敗の戻りコードのみ必要 (応答ビジネス・オブジェクトは不要)	カスタム・プロトコル・ハンドラー (*31 ページの『第 4 章 カスタム・プロトコル・ハンドラーの作成』*を参照)

表 5. WebSphere Business Integration Adapter コンポーネントの使用あるいはカスタム・コンポーネントの作成 (続き)

WebSphere Business Integration Adapter 提供のコンポーネント	左記コンポーネントを使用できる条件 (すべてが満たされること)	作成が必要なカスタム・コンポーネント
XML データ・ハンドラー	データ形式は XML 1.0 (「データ・ハンドラー・ガイド」を参照)	カスタム・データ・ハンドラー (「データ・ハンドラー・ガイド」を参照)
名前リゾルバー (XML データ・ハンドラー)	ビジネス・オブジェクト名は、XML 文書のルート・エレメント名と、XML データ・ハンドラーの子メタオブジェクトの `BOPrefix` 属性によって決定します (構成可能)。	カスタム名前リゾルバー (「データ・ハンドラー・ガイド 」を参照)
エンティティー・リゾルバー (XML データ・ハンドラー)	• 外部エンティティーは無視 • ローカル・ファイル・システムから外部エンティティーを検索	カスタム・エンティティー・リゾルバー (「データ・ハンドラー・ガイド」を参照)
SAX パーサー (XML データ・ハンドラー)	データ形式は XML	カスタム・パーサー

コネクター・ビジネス・オブジェクトの処理

コネクターは、統合ブローカーとプロトコル・ハンドラーとの間で、ビジネス・オブジェクトをやりとりします。コネクター・エージェントはプロトコル・ハンドラーに要求ビジネス・オブジェクトを送り、プロトコル・ハンドラーから応答ビジネス・オブジェクトを受け取ります。ただし、ビジネス・オブジェクトに含まれるデータは処理の対象にはなりません。

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渡すと、コネクターは次の処理を実行します。

1. トップレベルのビジネス・オブジェクトから要求ビジネス・オブジェクトを抽出します。コネクターは、要求ビジネス・オブジェクトが最初の子ビジネス・オブジェクトであり、`CxIgnore` の値も `CxBlank` の値もとらないものと想定します。
2. 要求ビジネス・オブジェクトをプロトコル・ハンドラーに送ります。
3. プロトコル・ハンドラーから応答ビジネス・オブジェクトが戻ると、コネクターはこの応答ビジネス・オブジェクトをトップレベルのビジネス・オブジェクトに追加し、その結果としてのトップレベルのビジネス・オブジェクトを統合ブローカーに戻します。

コネクター・ビジネス・オブジェクトの構造

コネクターには階層を持つビジネス・オブジェクトが必要です。トップレベルのビジネス・オブジェクトには属性が格納されています。その値は、宛先 URL ストリング、データの MIME タイプ、ビジネス・オブジェクトのプレフィックス、さらには要求および応答のビジネス・オブジェクトです。

図 6 は、IBM WebSphere Business Integration Adapter for XML のトップレベルのビジネス・オブジェクトに必要な基本構造を示したものです。

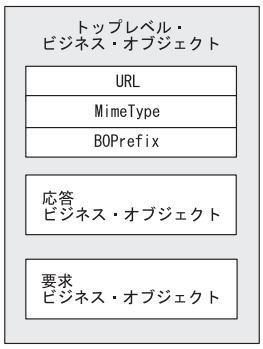

図 6. トップレベル・ビジネス・オブジェクトの基本構造

例えば、2 つのビジネス・オブジェクト XMLApp_CustCreateRequest および XMLApp_CustCreateResponse を作成する場合、コネクター用のトップレベルのビジネス・オブジェクト定義は次のようになります。

```
XMLApp_CustCreate

URL      String
MimeType String
BOPrefix String
Response XMLApp_CustCreateResponse
Request  XMLApp_CustCreateRequest
```

Business Object Designer を使用して、要求および応答のビジネス・オブジェクトを作成します。トップレベルのビジネス・オブジェクト定義を作成し、必須属性を要求および応答ビジネス・オブジェクトに追加します。次に、トップレベルのビジネス・オブジェクトをサポートするようにコネクターを構成します。

トップレベルのビジネス・オブジェクトの必須属性

トップレベルのビジネス・オブジェクトには、URL ストリング、MIME タイプ、BOPrefix、要求ビジネス・オブジェクト、および応答ビジネス・オブジェクトに対応した属性が 1 つ以上必要です。これらの各属性を `IsRequired = True` としてマークする必要があります。

表 6 に、トップレベルのビジネス・オブジェクトの必須属性の説明を示します。詳細については、以降のセクションを参照してください。

表 6. トップレベル XML ビジネス・オブジェクトの必須属性

属性	タイプ	説明
URL	ストリング	宛先の URL。
MimeType	ストリング	トランザクションに使用される MIME タイプ。
BOPrefix	ストリング	MIME タイプとともに、XML データ・ハンドラーのインスタンス作成に使用されます。
応答	ビジネス・オブジェクト	応答メッセージを表すビジネス・オブジェクト。27 ページの『要求ビジネス・オブジェクトと応答ビジネス・オブジェクト』を参照。
要求	ビジネス・オブジェクト	要求メッセージを表すビジネス・オブジェクト。トップレベルのビジネス・オブジェクトでは、応答ビジネス・オブジェクトに対応する属性の後に、この属性を設定してください。

注: コネクターでは、少なくとも 1 つの属性をキー属性として設定することが必要です。ただし、コネクターでは、キーとして設定された属性がなくても差し支えありません。

URL

URL ストリングは、ビジネス・オブジェクト内のデータの宛先と、データを渡すときに使用されるプロトコルを定義します。このストリングには、プロトコル (HTTP や HTTPS など) を含む宛先全体が設定されるため、プロトコルを指定する別の属性は必要ありません。

コネクターは、URL ストリングを使用して、宛先 URL との接続を開設します。接続が開設されると、コネクターは URL ストリングを使用して、適切なプロトコル・ハンドラーのインスタンスを作成します。

例えば、`http://www.ibm.com` というストリングは、HTTP プロトコルが使用されること、および HTTP プロトコル・ハンドラーのインスタンスが作成されることを指定します。

MIME タイプ

MIME タイプは、URL に渡されるデータの内容タイプと形式を定義します。コネクターは MIME タイプを使用して、適切なデータ・ハンドラーを起動します。メタオブジェクトは、その MIME タイプと BOPrefix の組み合わせを処理するデータ・ハンドラーのインスタンスを指定するものです。実装されたデータ・ハンドラーで処理できる MIME タイプが 1 つだけである場合、子メタオブジェクトでの BOPrefix 属性の指定は任意です。トップレベルのビジネス・オブジェクトの場合、BOPrefix は必須です。

特に指定しない限り、コネクターは MIME タイプを `text/xml` と見なしますが、コネクターの設定によって他の MIME タイプを使用することもできます。

BOPrefix

コネクターは、BOPrefix と MimeType 属性の組み合わせに基づいて適切なデータ・ハンドラーを起動します。この属性は、ビジネス・オブジェクト名の一意性を保証するために必要な属性です。例えば、2 つのアプリケーション AppA_PO および AppB_PO に対応して 2 つの仕入れ注文ビジネス・オブジェクトを設定できます。

注: トップレベルのビジネス・オブジェクトの BOPrefix 属性は、XML データ・ハンドラーに対応する子メタオブジェクトの BOPrefix 属性とは異なっています。XML データ・ハンドラーの子メタオブジェクトの詳細については、「*データ・ハンドラー・ガイド*」を参照してください。

URL から XML ストリームが戻ると、XML データ・ハンドラーは XML ストリームのルート・エレメント名を、ビジネス・オブジェクト定義 `BOPrefix_name` にマップします。ルート・エレメント名の値は、必ず BOPrefix の値の後に置かれます。

例えば、XML 文書のルート・エレメントが <Customer> で、`BOPrefix=AppA` の場合、`BOPrefix_Name` は AppA_Customer となります。

要求ビジネス・オブジェクトと応答ビジネス・オブジェクト

要求と応答のビジネス・オブジェクトには、宛先 URL に渡される実際のデータが格納されています。コネクターがトップレベルのビジネス・オブジェクトを受け取ると、要求ビジネス・オブジェクトのみが取り込まれます。応答ビジネス・オブジェクトは、宛先 URL から戻されたデータとともに取り込まれます。

トップレベルのビジネス・オブジェクト内の要求および応答のビジネス・オブジェクトを定義するとき、次のガイドラインに従ってください。

- 以下の条件が成立する場合、応答ビジネス・オブジェクトは要求ビジネス・オブジェクトの前に置きます。
  - 要求ビジネス・オブジェクトと応答ビジネス・オブジェクトが同じタイプ。
  - ビジネス・オブジェクトはコラボレーションの要求に使用される (統合ブローカーが WebSphere ICS である場合のみ)。
  - 要求ビジネス・オブジェクト内のデータは保存が必要 (URL からの応答による上書きは不可)。
- トップレベルのビジネス・オブジェクトで、応答ビジネス・オブジェクトに対応する属性値は、`CxIgnore` または `CxBlank` に設定します。コネクターは、最初の null でない属性値をプロトコル・ハンドラーに渡します。
- 要求を表すビジネス・オブジェクトが応答を表すビジネス・オブジェクトと同一である場合、要求属性のタイプと応答属性のタイプは同じです。
- 要求と応答のビジネス・オブジェクトは異なる場合があります。例えば、顧客からの仕入れ注文ビジネス・オブジェクトを送り、注文状況のビジネス・オブジェクトを受け取ることなどが考えられます。
- トップレベルのビジネス・オブジェクトに戻される各応答 XML 文書をサポートするために、複数の応答ビジネス・オブジェクトを定義することができます。コネクターは、複数の応答ビジネス・オブジェクトを使用することにより、異なるタイプの XML 文書 (異なるタイプのビジネス・オブジェクトに対応) が Web サーバーから戻る可能性があっても、その状況に対処できます。

ビジネス・オブジェクトのデータ・ハンドラー要件への準拠

コネクターに対応するトップレベルのラッパー・ビジネス・オブジェクトには任意の WebSphere Business Integration Adapter ビジネス・オブジェクトを格納できますが、格納されたビジネス・オブジェクトのデータは、データの変換に使用されるデータ・ハンドラーの要件に準拠した形式でデリバリーされることが必要です。

例えば、BySize データ・ハンドラーの場合、ビジネス・オブジェクト定義は、各ビジネス・オブジェクトの MaxLength 属性プロパティーに対して値を指定することが必要です。XML データ・ハンドラーの場合、ビジネス・オブジェクト定義には、データ・ハンドラーによる XML 文書の生成を可能にするアプリケーション固有情報が必要です。

したがって、処理されるデータの型ごとに、専用のビジネス・オブジェクトを作成しておくことが望ましいやり方です。ビジネス・オブジェクトには、アプリケーションが必要とするデータとデータ・ハンドラーが必要とする情報のみを設定します。こうして作成されたビジネス・オブジェクトをトップレベルのコネクター・ビジネス・オブジェクトに格納することができます。

各データ・ハンドラーに固有の詳細については、「データ・ハンドラー・ガイド」を参照してください。

イベント通知用のビジネス・オブジェクト

イベント通知ビジネス・オブジェクトの構造は、要求ビジネス・オブジェクトの構造と似ています。どちらも、URL、MIME タイプ、BOPrefix、応答ビジネス・オブジェクト、および要求ビジネス・オブジェクトに対応する属性が必要です。ビジネス・オブジェクトの処理における唯一の相違は、コネクターが応答ビジネス・オブジェクトの内容を取り扱う方法です。イベント通知の場合、コネクターは、応答ビジネス・オブジェクトに、イベントを表す子ビジネス・オブジェクトが格納されていると想定しています。

イベント通知ビジネス・オブジェクトを定義するときには、次の点に留意してください。

- トップレベルのビジネス・オブジェクトには、要求と応答の両方の属性が必要です。要求と応答の属性は両方とも必須 にする必要があり、異なるタイプの属性にする必要があります。
- 要求ビジネス・オブジェクトは応答ビジネス・オブジェクトの前に置きます。
- 応答ビジネス・オブジェクトは、同じタイプの複数の子ビジネス・オブジェクトを戻すことができます。例えば、顧客イベントのみを戻す応答ビジネス・オブジェクトを設計できます。
- 応答ビジネス・オブジェクトは、異なるタイプの複数の子ビジネス・オブジェクトを戻すことができます。例えば、注文イベントと顧客イベントのみを戻す応答ビジネス・オブジェクトを設計できます。
- アンサブスクライブされた子ビジネス・オブジェクトはすべてアーカイブ・ディレクトリーにアーカイブされます。
- ビジネス・オブジェクトでは、その定義のアプリケーション固有情報列に指定されたデフォルトの動詞の他に、サポートされている動詞の列に、「DefaultVerbName」動詞が追加されている必要があります。デフォルトの動詞は、サブスクリプションが正しくチェックできるように、イベント通知のために使用される動詞です。統合ブローカーに送られるビジネス・オブジェクトごとに、動詞を設定することが必要です。

図 7 に、ビジネス・オブジェクト定義における「DefaultVerbName」の配置を示します。

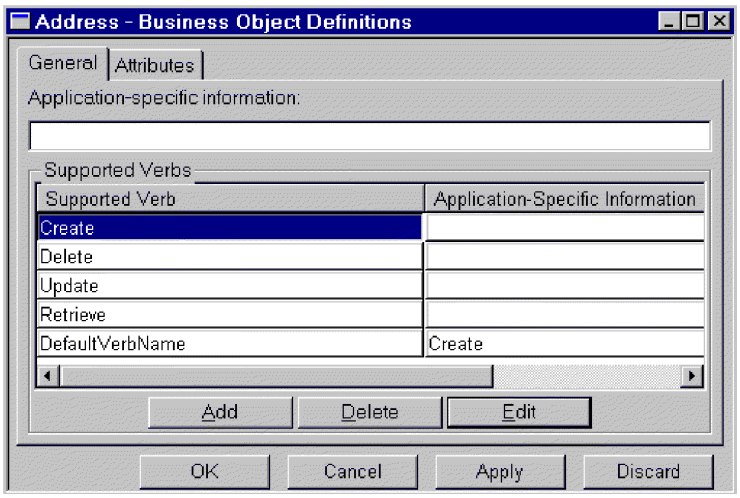

図 7. ビジネス・オブジェクト定義における「DefaultVerbName」の配置

XML DTD またはスキーマ文書に基づくビジネス・オブジェクト

XML DTD またはスキーマ文書に基づいて要求ビジネス・オブジェクトと応答ビジネス・オブジェクトを作成する場合、処理される XML 文書のタイプごとに、ビジネス・オブジェクト定義を作成する必要があります。ビジネス・オブジェクト定義には、XML 文書の DTD またはスキーマ文書に含まれる構造情報が含まれています。例えば、1 つの要求ストリーム (1 つの DTD またはスキーマ文書) があり、一方 4 つの応答ストリーム・タイプ (4 つの異なる DTD またはスキーマ文書) が存在する可能性がある場合、5 つのビジネス・オブジェクト定義を作成する必要があります。これに対し、要求ストリームと応答ストリームが同じスキーマを使用する場合には、必要なビジネス・オブジェクト定義は 1 つのみです。XML Object Discovery Agent (ODA) を使用すると、DTD またはスキーマ文書に基づいてビジネス・オブジェクト定義を生成することができます。

注: DTD またはスキーマの読み取り中は、XML ODA は FIXED 属性を無視します。これは、XML インスタンスにおいてこれらの属性の値はオプションであり、値が常に固定されているためです。BO より作成され、BO に読み込まれる XML インスタンス内にこれらの FIXED 値が含まれるようにするには、FIXED 属性を BO 属性として手動で追加する必要があります。実行時にこれらの値が変更されないよう、確認する必要があります。

XML ODA を介してまたは手動で XML 文書用のビジネス・オブジェクト定義を生成する方法の詳細については、「データ・ハンドラー・ガイド」を参照してください。

第 4 章 カスタム・プロトコル・ハンドラーの作成

この章ではプロトコル・ハンドラー・フレームワークと、それを使用してカスタム・プロトコル・ハンドラーを作成する方法について説明します。次のセクションが含まれています。

プロトコル・ハンドラー・フレームワーク

開発者は、WebSphere Business Integration Adapter のプロトコル・ハンドラー・フレームワークを使用することにより、各種のプロトコルに対応したプロトコル・ハンドラーを、統一された方法に従って作成することができます。プロトコル・ハンドラー・フレームワークには、CWURLConnection というクラスがあり、このクラスにはカスタム・プロトコル・ハンドラーの作成時に実装を必要とする抽象メソッドが所属しています。このフレームワークは、com.crossworlds.protocolhandler パッケージの一部です。

プロトコル・ハンドラー・フレームワークのクラス

各カスタム・プロトコル・ハンドラーは、少なくとも次の 2 つのクラスを持つ必要があります。

- Handler
- cw_*protocol*connection (HTTP プロトコルでは cw_httpconnection)

connection クラスは、CWURLConnection クラスを拡張します。

図 8 に、com.crossworlds.connectors.utils.protocolhandler 基本クラスの階層を示します。

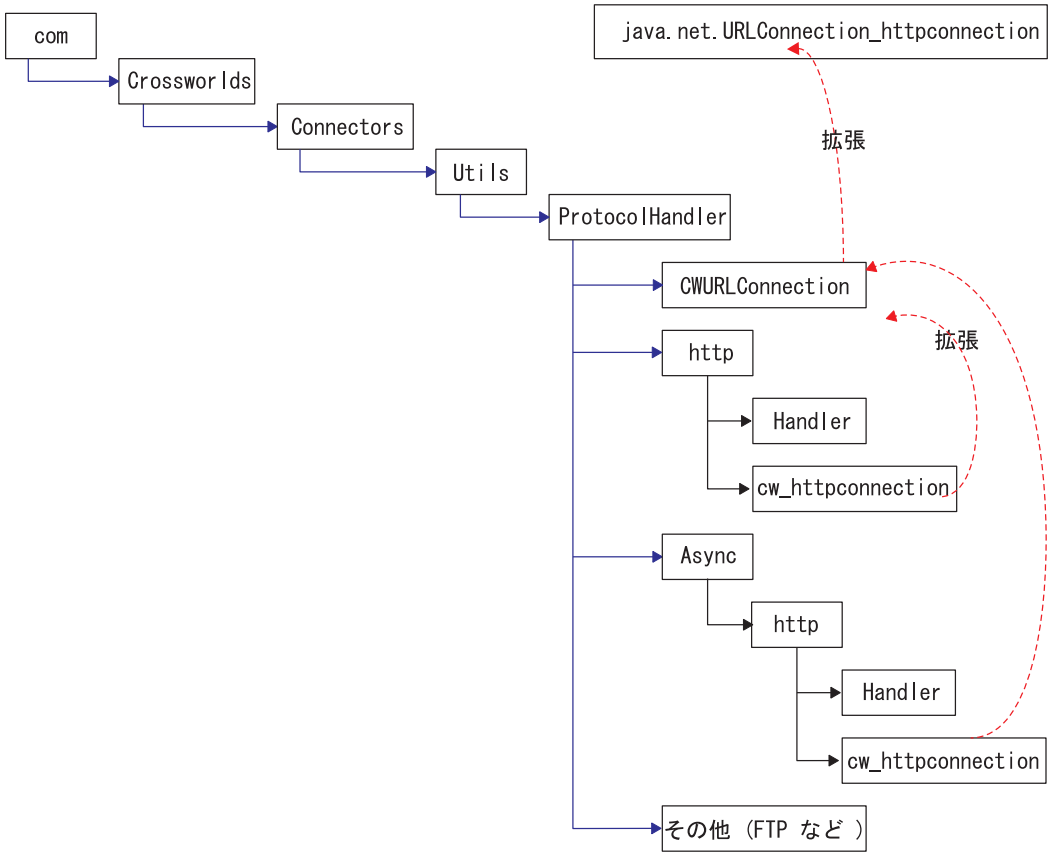

図 8. プロトコル・ハンドラーのクラス階層

プロトコル・ハンドラー・フレームワークを使用してカスタム・プロトコル・ハンドラーを開発する手順は次のとおりです。

- ProtocolNameConnection クラスを作成します。ここで、ProtocolName は使用するプロトコルの名前です。
- connection クラス内の getContent() メソッドの 1 つ以上の実装を提供します。
- Handler クラスを作成します。

Handler クラスのサマリー

```
Public URLConnection openConnection(URL url); throws IOException
```

Connection クラスのサマリー

```
public String getContent (object input, String mimeType,
Sting BOPrefix, Long ServerTimeout, String Credentials) throws IOException
public String getContentType()
public synchronized void connect() throws IOException
```

Protocol Handler クラスの作成

コネクターをインストールすると、プロトコル・ハンドラー用のスタブ・コード・ファイルと Make ファイルがインストールされます。スタブ・ファイルには、実装を必要とするすべてのメソッドを列挙した空のクラスを定義している Java コードが格納されています。このスタブ・ファイルをテンプレートとして使用して、カスタム・プロトコル・ハンドラーを作成することができます。

新しいプロトコル・ハンドラーを実装するには、次の手順を実行します。

1. stubProtocolHandler.java ファイルを変更 (および名前変更) します。
2. Make ファイルを編集して、ソース・ファイルの名前を設定します。
3. makeProtocolHandler.bat ファイルを実行して、クラスをコンパイルします。Make ファイルはクラスのコンパイルのみを実行します。クラスは .jar ファイルには追加されません。
4. 新規のクラスを .jar ファイルに追加します。次のコマンドを使用します。

   jar cvf*MyProtocolHandler*.jar <*classes*>

   ここで、以下のように説明されます。

   - MyProtocolHandler.jar は、プロトコル・ハンドラー .jar ファイルです。このファイルは、コネクターの始動バッチ・ファイル start_xml.bat (UNIX では start_xml.sh) が存在するクラスパスに設定されていることが必要です。
   - <classes> は、使用するプロトコル・ハンドラーのすべてのクラスです。すべてのクラスを列挙し、各クラスに該当する項目をスペースで区切ります。
5. コネクターにより新規のクラスが選択されることを確認します。start_xml.bat (UNIX では start_xml.sh) を編集して、新規の .jar ファイルが CLASSPATH に含まれるようにします。

プロトコル・ハンドラー・フレームワークのメソッド

次のセクションでは、新しいプロトコル・ハンドラーを設計したり、既存のプロトコル・ハンドラーを修正するときに使用するプロトコル・ハンドラー・フレームワークのメソッドについて説明します。

getContent ()

getContent() メソッドは、ビジネス・オブジェクト処理用に使用されます。このメソッドの実行内容は次のとおりです。

- MimeType と BOPrefix ビジネス・オブジェクト属性から、作成するデータ・ハンドラーのインスタンスを決定します。
- 変換のために適したデータ・ハンドラーにビジネス・オブジェクトを送り、次にファイルを URL に送ります。
- 宛先 URL から応答ストリームを受け取り、このストリームを WebSphere Business Integration Adapter ビジネス・オブジェクトに変換するため、データ・ハンドラー・インスタンスを起動します。
- ビジネス・オブジェクトを当初の呼び出し元 (コネクターなど) に戻します。

構文

```
public abstract Object getContent (Object input, String mimeType,
String BOprefix,Long ServerTimeoutStringCredentials) throws IOException
```

パラメーター

input ビジネス・オブジェクト・インターフェース (送信対象のビジネス・オブジェクト) を指定します。

mimeType データ・ハンドラーに渡すデータの MIME タイプを指定します。

BOprefix データ・ハンドラーに渡すデータの BOPrefix を指定します。

Long ServerTimeout
タイムアウトになるまでにサーバーが待機する時間を指定します。

Credentials
認証クリデンシャル。

戻り値

ビジネス・オブジェクト・インターフェースを戻します。

WebSphere Business Integration Adapter 提供のプロトコル・ハンドラーの呼び出し

次のコードに、WebSphere Business Integration Adapter 提供のプロトコル・ハンドラーを呼び出す方法を示します。

```
try
{
// set the system property, so that Java knows where to look for
// the protocol handlers. You only need to do it once.
Properties prop = System.getProperties();
prop.put("java.protocol.handler.pkgs",
"com.crossworlds.connectors.utils.ProtocolHandlers");

URL url = new URL("http://www.crossworlds.com");
CWURLConnection uc = (CWURLConnection) url.openConnection();
BusinessObjectInterface respBO = (BusinessObjectInterface)
uc.getContent (input, mime, prefix, Long ServerTimeout, Credentials);

}
catch (Exception XX)
{
//flag error
}
```

カスタム・プロトコル・ハンドラーのサンプル・コード

カスタム・プロトコル・ハンドラーを開発するとき、次のサンプル・コードが手引きとして役に立ちます。

```
/**
 * This package hierarchy is used to write the Protocol Handler.
 * [ProtocolName] should be substituted with the name of the protocol
 * for which the handler is being written.
 * For example com.crossworlds.connectors.utils.ProtocolHandlers.ftp
   or com.crossworlds.connectors.utils.ProtocolHandlers.http
 */
package com.crossworlds.connectors.utils.ProtocolHandlers.[ProtocolName];


import CxCommon.BusinessObjectInterface;
import com.crossworlds.connectors.utils.ProtocolHandlers.CWURLConnection;
import com.crossworlds.DataHandlers.DataHandler;

import AppSide_Connector.JavaConnectorUtil;

import java.net.*;
import java.io.*;

/**
 * The handler class creates a ProtocolNameConnection class instance
 * It is invoked indirectly via Java's URL getContent() mechanism.
 *
 * how to use it:
 * System.setProperty ("java.protocol.handler.pkgs",
 *"com.crossworlds.ProtocolHandler");
 *URL url = new URL ("the URL");
 *CWURLConnection uc  = (CWURLConnection) url.openConnection ();
* /
public class Handler
{
   // this will return the appropriate URLConnection
   // But the constructor takes only one argument - the URL.  As this
   // is called by Java Networking Framework.
   public URLConnection openConnection(URL url) throws IOException
   {
      // you can pass in any parameters here.
        return new MyURLConnection (url);
   }
}

class MyURLConnection extends CWURLConnection {

   /**
    * This is instantiated by URL.openConnection()
    */
   public MyURLConnection(URL url)
   {
      // store this URL some where
   }

   /**
    * This method returns the content type of the data
    */
   public String getContentType()
   {
      // here is where you have to determine the content Type (aka
      // Mimetype) of URL streams
   }

   /**
    *  This method is used to create a connection
```

```
     */
    public synchronized void connect() throws IOException
    {
       // you might call super().connect as it suffices most of the
       // time.
       // If it is custom protocol, do the handshaking stuff here
    }

    /**
     * getContent () : The getContent method used by CrossWorlds.
     * This method takes in 5 parameters
     *   - input Object,
     *   - content type for the data &
     *   - Business Object Prefix to * be used to create the Business
     *   Object name
     *       - Long ServerTimeout
     *   - Credentials
     *   It returns an appropriate Object back to the caller.  This
     *   method interacts with the DataHandler using the exposed APIs
     *   for the DataHandler.
     */
    public Object getContent(Object input,  String mimeType,
  String BOprefix, Long ServerTimeout, String Credentials)
                                  throws IOException
    {
       // log a message
       JavaConnectorUtil.logMessage
       ("logging a message", JavaConnectorUtil.XRD_INFO);

       // write a trace
       if (JavaConnectorUtil.isTraceEnabled (JavaConnectorUtil.LEVEL3))
          JavaConnectorUtil.traceWrite  (JavaConnectorUtil.LEVEL3,
          "Level 3 trace msg");


       // get a datahandler
       DataHandler dh = DataHandler.createHandler (null, mimeType, BOprefix);

       InputStream in = dh.getStreamFromBO
       ( (BusinessObjectInterface) input, null);

       // Send this to URL
          - read data from Input Stream
          - write to URL
          - repeat until input stream is drained.

       // Now read the response
       String replyString = // some how read the reply from URL
       String outputType = // get the mime of reply some how

// remember to get a fresh DH, as the incoming data may be of
// different mime type than was originally received by the
// protocol handler
       DataHandler dh2 = DataHandler.createHandler
       (null, outputType, BOprefix);

       BusinessObjectInterface replyBO = dh2.getBO
       (replyString, outputType);

       return replyBO;  // DONE!

    }

}
```

付録 A. コネクターの標準構成プロパティー

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

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

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

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

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

新規プロパティーと削除されたプロパティー

本リリースには、次の標準プロパティーが追加されました。

新規プロパティー

- XMLNameSpaceFormat

削除されたプロパティー

- RestartCount

標準コネクター・プロパティーの構成

アダプター・コネクターには 2 つのタイプの構成プロパティーがあります。

- 標準構成プロパティー
- コネクター固有のプロパティー

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

Connector Configurator の使用

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

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

プロパティー値の設定と更新

プロパティー・フィールドのデフォルトの長さは 255 文字です。

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

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

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

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

特定のプロパティーの更新方法を確認するには、「Connector Configurator」ウィンドウ内の**「更新メソッド」**列を参照するか、次に示す 39 ページの表 7 の「更新メソッド」列を参照してください。

標準プロパティーの要約

表 7 は、標準コネクター構成プロパティーの早見表です。標準プロパティーの依存関係は RepositoryDirectory に基づいているため、コネクターによっては使用されないプロパティーがあり、使用する統合ブローカーによってプロパティーの設定が異なる可能性があります。

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

注: 表 7 の「注」列にある「Repository Directory は REMOTE」という句は、ブローカーが InterChange Server であることを示します。ブローカーが WMQI または WAS の場合には、リポジトリー・ディレクトリーは LOCAL に設定されます。

表 7. 標準構成プロパティーの要約

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

表 7. 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
ControllerStoreAndForwardMode	true または false	true	動的	Repository Directory は <REMOTE> (ブローカーは ICS)
ControllerTraceLevel	0 から 5	0	動的	Repository Directory は <REMOTE> (ブローカーは ICS)
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	コンポーネント再始動	JMS トランスポートのみ
DeliveryTransport	MQ、IDL、または JMS	JMS	コンポーネント再始動	Repository Directory がローカルの場合は、値は JMS のみ
DuplicateEventElimination	true または false	false	コンポーネント再始動	JMS トランスポートのみ: Container Managed Events は <NONE> でなければならない
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory または CxCommon.Messaging.jms.SonicMQFactory または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	FactoryClassName が IBM の場合は crossworlds.queue.manager を使用。FactoryClassName が Sonic の場合 localhost:2506 を使用。	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ

表 7. 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
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	Repository Directory は <REMOTE> (ブローカーは ICS)
JvmMaxNativeStackSize	スタックのサイズ (キロバイト単位)	128k	コンポーネント再始動	Repository Directory は <REMOTE> (ブローカーは ICS)
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	Repository Directory は <REMOTE> (ブローカーは ICS)
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならない
Locale	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR **注:** これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	true または false	false	コンポーネント再始動	Repository Directory は <REMOTE> でなければならない (ブローカーは ICS)
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならない (ブローカーは ICS)
MessageFileName	パスまたはファイル名	CONNECTORNAMEConnector.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	*CONNECTORNAME*/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は true でなければならない

表 7. 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
OADAutoRestartAgent	`true` または `false`	`false`	動的	Repository Directory は <REMOTE> でなければならない (ブローカーは ICS)
OADMaxNumRetry	正数	`1000`	動的	Repository Directory は <REMOTE> でなければならない (ブローカーは ICS)
OADRetryTimeInterval	正数 (単位: 分)	`10`	動的	Repository Directory は <REMOTE> でなければならない (ブローカーは ICS)
PollEndTime	`HH:MM`	`HH:MM`	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒) `no` (ポーリングを使用不可にする) `key` (コネクターのコマンド・プロンプト・ウィンドウで文字 `p` が入力された場合にのみポーリングする)	`10000`	動的	
PollQuantity	`1` から `500`	`1`	エージェント再始動	JMS トランスポートのみ: Container Managed Events を指定
PollStartTime	`HH:MM` (HH は 0 から 23、MM は 0 から 59)	`HH:MM`	コンポーネント再始動	
RepositoryDirectory	メタデータ・リポジトリーの場所		エージェント再始動	ICS の場合は <REMOTE> に設定する。WebSphere MQ Message Brokers および WAS の場合: C:¥crossworlds¥repository に設定する

表 7. 標準構成プロパティーの要約 (続き)

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

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

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

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

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

AgentConnections プロパティーは、orb.init[] により開かれる ORB (オブジェクト・リクエスト・ブローカー) 接続の数を制御します。

このプロパティーのデフォルト値は 1 に設定されます。必要に応じてこの値を変更できます。

AgentTraceLevel

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

ApplicationName

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

BrokerType

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

CharacterEncoding

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

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

デフォルトでは、ドロップダウン・リストには、サポートされる文字エンコードの一部のみが表示されます。ドロップダウン・リストに、サポートされる他の値を追加するには、製品ディレクトリーにある ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。詳細については、本書の Connector Configurator に関するセクションを参照してください。

ConcurrentEventTriggeredFlows

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

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

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

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

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

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

ContainerManagedEvents

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

デフォルト値はありません。

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

- PollQuantity = 1 から 500
- SourceQueue = /SOURCEQUEUE

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

これらのプロパティーはアダプター固有ですが、**例**の値は次のようになります。

- 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 を JMS に設定した場合にのみ表示されます。

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

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

ControllerStoreAndForwardMode

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

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

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

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

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

デフォルトは true です。

ControllerTraceLevel

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

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

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

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

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

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

WebSphere MQ および IDL

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

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

JMS

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

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

このプロパティーは、false に設定することもできます。

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

FaultQueue

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

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

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

デフォルト値は 128M です。

JvmMaxNativeStackSize

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

デフォルト値は 128K です。

JvmMinHeapSize

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

デフォルト値は 1M です。

jms.FactoryClassName

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

デフォルトは CxCommon.Messaging.jms.IBMMQSeriesFactory です。

jms.MessageBrokerName

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

デフォルトは 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。

リモート・メッセージ・ブローカーに接続すると、このプロパティーは次の (必須) 値をとります。
QueueMgrName:<Channel>:<HostName>:<PortNumber>
各変数の意味は以下のとおりです。
QueueMgrName: キュー・マネージャー名です。
Channel: クライアントが使用するチャネルです。
HostName: キュー・マネージャーの配置先のマシン名です。
PortNumber: キュー・マネージャーが listen に使用するポートの番号です。

例えば、次のように指定します。

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

jms.NumConcurrentRequests

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

デフォルト値は 10 です。

jms.Password

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

デフォルトはありません。

jms.UserName

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

デフォルトはありません。

ListenerConcurrency

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

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

Locale

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

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、以下のように説明されます。

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

デフォルトでは、ドロップダウン・リストには、サポートされるロケールの一部のみが表示されます。ドロップダウン・リストに、サポートされる他の値を追加する

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

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

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

LogAtInterchangeEnd

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

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

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

MaxEventCapacity

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

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

MessageFileName

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

MonitorQueue

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

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

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

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

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

デフォルト値は false です。

OADMaxNumRetry

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

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

デフォルト値は 1000 です。

OADRetryTimeInterval

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

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

デフォルトは 10 です。

PollEndTime

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

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

PollFrequency

これは、前回のポーリングの終了から次のポーリングの開始までの間の間隔です。PollFrequency 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 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。一部のアダプターでは、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 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。

PollFrequency は以下の値のいずれかに設定します。

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

デフォルトは 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

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

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

コネクターは、1 回目のポーリングを受けると、メッセージの本文を選出します。これは、本文が添付とも見なされるからです。本文の MIME タイプにはデータ・ハンドラーが指定されていないので、コネクターは本文を無視します。
コネクターは PO の最初の添付を処理します。この添付の MIME タイプには対応する DH があるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
2 回目のポーリングを受けると、コネクターは PO の 2 番目の添付を処理します。この添付の MIME タイプには対応する DH があるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
これが受け入れられると、PO の 3 番目の添付が届きます。

PollStartTime

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

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

RequestQueue

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

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

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

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

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

ResponseQueue

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

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

RestartRetryCount

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

デフォルトは 3 です。

RestartRetryInterval

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

デフォルトは 1 です。

RHF2MessageDomain

WebSphere Message Brokers および WAS のみ

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

サンプル・ヘッダーを次に示します。

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

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

SourceQueue

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

JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、45 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

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

WsifSynchronousRequestTimeout

WAS 統合ブローカーでのみ使用されます。

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

デフォルト値は 0 です。

XMLNameSpaceFormat

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

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

デフォルト値は short です。

付録 B. Connector Configurator

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

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

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

注:

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

この付録では、次のトピックについて説明します。

Connector Configurator の概要

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

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

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

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

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

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

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

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

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

ただし**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクター固有プロパティーのテンプレートを作成します。システム内で既にテンプレートが作成されている場合には、作成されているテンプレートを使用します。システム内でまだテンプレートが作成されていない場合には、60 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

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

Connector Configurator の始動

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

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

スタンドアロン・モードでの Configurator の実行

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

これを行うには、以下のステップを実行します。

- **「スタート」>「プログラム」**から、**「IBM WebSphere InterChange Server」>「IBM WebSphere Business Integration Tools」>「Connector Configurator」**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ます。使用しているブローカーに応じて、ICS、WebSphere Message Brokers、または WAS を選択します。

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (65 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

- 「System Manager」ウィンドウの「コネクター」フォルダーで構成ファイルを選択し、右クリックします。Connector Configurator が開き、この構成ファイルの統合ブローカー・タイプおよびファイル名が上部に表示されます。
- Connector Configurator 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、 60 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でこのテンプレートを保管します。既存のテンプレートは ¥WebSphereAdapters¥bin¥Data¥App ディレクトリーにあります。

新規テンプレートの作成

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

Connector Configurator 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 (Connector-Specific Property Template)」**とクリックします。
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、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
  更新されたメソッド
  説明
- **フラグ**
  標準フラグ
- **カスタム・フラグ**
  フラグ

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

値の指定

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

1. **「値」**タブをクリックします。「一般」のパネルに代わって「値」の表示パネルが表示されます。
2. **「プロパティーを編集」**表示でプロパティーの名前を選択します。
3. **「最大長」**および**「最大複数値」**のフィールドに値を入力します。

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. **「プロパティーを編集」**リストでプロパティーを選択し、右マウス・ボタンでクリックします。
2. ダイアログ・ボックスから**「追加」**を選択します。
3. 新規プロパティー値の名前を入力し、「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、作成した以前の値が表示されます。

「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」**表示で強調表示された依存プロパティーで、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 (Finish)」**をクリックします。Connector Configurator により、XML 文書として入力した情報が、Connector Configurator がインストールされている ¥bin ディレクトリーの ¥data¥app の下に保管されます。

新しい構成ファイルを作成

構成ファイルを新規に作成するには、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。

- 「System Manager」ウィンドウで**「コネクター」**フォルダーを右クリックし、**「新規コネクターの作成」**を選択します。Connector Configurator が開き、**「新規コネクター」**ダイアログ・ボックスが表示されます。
- スタンドアロン・モードの場合は、Connector Configurator で**「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 接続、WebSphere Message Brokers 接続、WAS 接続のいずれかを選択します。
- この章で後述する説明に従って**「新規コネクター」**ウィンドウの残りのフィールドに入力します。

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、そのテンプレートを使用して構成ファイルを作成できます。

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

- **名前**

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

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

- **システム接続**

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

- **「コネクター固有プロパティー・テンプレート (Connector-Specific Property Template)」を選択します。**

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

  使用するテンプレートを選択し、**「OK」**をクリックします。

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

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

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

- コネクター定義ファイル。
  コネクター定義ファイルは、特定のコネクターのプロパティーと、適用可能なデフォルト値がリストされたテキスト・ファイルです。コネクターの配布パッケー

ジの ¥repository ディレクトリー内には、このようなファイルが格納されていることがあります (通常、このファイルの拡張子は .txt です。例えば、XML コネクターの場合は CN_XML.txt です)。

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

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

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

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

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

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

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

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

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

構成ファイルの完成

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

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

1. **「標準のプロパティー」**タブで、BrokerType プロパティーの値フィールドを選択します。ドロップダウン・メニューで、値 ICS、WMQI、または WAS を選択します。
2. 選択したブローカーに関連付けられているプロパティーが「標準のプロパティー」タブに表示されます。ここでファイルを保管するか、または 68 ページの『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。
3. 構成が完了したら、**「ファイル」>「保管」>「プロジェクトに」**を選択するか、または**「ファイル」>「保管」>「ファイルに」**を選択します。

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

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

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

構成ファイル・プロパティーの設定

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

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

- 標準のプロパティー
- コネクター固有のプロパティー
- サポートされるビジネス・オブジェクト
- トレース/ログ・ファイルの値

- データ・ハンドラー (保証付きイベント・デリバリーで JMS メッセージングを使用するコネクターの場合に該当する)

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

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

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

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

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

- コネクターの標準プロパティーは、コネクターのアプリケーション固有のコンポーネントとブローカー・コンポーネントの両方によって共用されます。すべてのコネクターが同じ標準プロパティーのセットを使用します。これらのプロパティーの説明は、各アダプター・ガイドの付録 A にあります。変更できるのはこれらの値の一部のみです。
- アプリケーション固有プロパティーは、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。

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

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

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下の手順を実行します。

1. 値を設定するフィールド内でクリックします。
2. 値を入力するか、ドロップダウン・メニューが表示される場合にはメニューから値を選択します。
3. 標準のプロパティーの値をすべて入力すると、以下のいずれかを実行することができます。
   - 変更内容を破棄し、元の値を保持したままで Connector Configurator を終了するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「いいえ」**をクリックします。
   - Connector Configurator 内の他のカテゴリーの値を入力するには、そのカテゴリーのタブを選択します。**「標準のプロパティー」** (またはその他のカテゴリー) で入力した値は、次のカテゴリーに移動しても保持されます。ウィンドウを閉じるときに、すべてのカテゴリーで入力した値を一括して保管するかまたは破棄するかを確認するプロンプトが出されます。
   - 修正した値を保管するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「はい」**をクリックします。「ファイル」メニューまたはツールバーから**「保管」>「ファイルに」**をクリックする方法もあります。

アプリケーション固有の構成プロパティーの設定

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

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

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

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

コネクター・プロパティーの暗号化

「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、暗号化が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・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 38 ページの『プロパティー値の設定と更新』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

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

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

ご使用のブローカーが ICS の場合

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

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

1. **「ビジネス・オブジェクト名」**リストの空のフィールドをクリックします。System Manager プロジェクトに存在するすべてのビジネス・オブジェクト定義を示すドロップダウン・リストが表示されます。
2. 追加するビジネス・オブジェクトをクリックします。
3. ビジネス・オブジェクトの**「エージェント・サポート」**(以下で説明) を設定します。

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ご使用のブローカーが WAS の場合

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

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

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

関連付けられているマップ (ICS のみ)

各コネクターは、現在 WebSphere InterChange Server でアクティブなビジネス・オブジェクト定義、およびそれらの関連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、サポートされるビジネス・オブジェクト

を追加指定した場合、それらの内容は、「Connector Configurator」ウィンドウの**「ファイル」**メニューから**「プロジェクトに保管」**を選択して、変更を保管した後に、このリストに反映されます。

- **関連付けられたマップ**

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

- **明示的**

  場合によっては、関連マップを明示的にバインドすることが必要になります。

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

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

  以下の手順を実行して、マップを明示的にバインドします。

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

リソース (ICS)

「リソース」タブでは、コネクター・エージェントがコネクター・エージェント並列処理を使用して、同時に複数のプロセスを処理するかどうか、またどの程度処理するかを決定する値を設定することができます。

すべてのコネクターでこの機能がサポートされるわけではありません。複数のプロセスを使用するよりも複数のスレッドを使用する方が通常は効率的であるため、Java でマルチスレッドとして設計されたコネクター・エージェントを実行している場合、この機能を使用することはお勧めできません。

メッセージング (ICS)

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

トレース/ログ・ファイル値の設定

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

ログとトレースの値を変更するには、以下の手順を実行します。

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

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

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

データ・ハンドラー

データ・ハンドラー・セクションの構成が使用可能となるのは、DeliveryTransport の値に JMS を、また ContainerManagedEvents の値に JMS を指定した場合のみです。すべてのアダプターでこのデータ・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 `ContainerManagedEvents` の下の説明を参照してください。その他の詳細は、「*コネクター開発ガイド (C++ 用)*」または「*コネクター開発ガイド (Java 用)*」を参照してください。

構成ファイルの保管

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

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

- System Manager から、統合コンポーネント・ライブラリーに *.con 拡張子付きファイルとして保管します。
- System Manager から、指定したディレクトリーに *.con 拡張子付きファイルとして保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに *.cfg 拡張子付きファイルとして保管します。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。
- WebSphere Application Server プロジェクトをセットアップしている場合には、このファイルを WebSphere Application Server プロジェクトに保管することもできます。

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

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

構成ファイルの変更

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

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

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

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

構成の完了

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

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

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

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

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

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

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

例えば 「Locale」 プロパティーの値のリストにロケール en_GB を追加するには、stdConnProps.xml ファイルを開き、以下に太字で示される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. XML Adapter のサンプル・シナリオの概要

ある企業で、IBM WebSphere Business Integration Adapter for XML を使用して、XML 文書を Web サーバーから読み取ったり、Web サーバーに POST する必要があるとします。このような場合に有効な 2 種類のサンプル・シナリオを以下に示します。これらのサンプル・シナリオは、簡単で、かつ XML Adapter の機能の基本的なポイントを示すように設計されています。

- WebSphere MQ Integrator Broker 接続での XML サンプル・シナリオのインストール

  このシナリオ例では、2 方向のデータ交換を表す 2 つの統合が必要です。

  – 最初の統合では、まず「XML_REQUEST_Order」イベントが WebSphere MQ Integrator Broker から WebSphere MQ キューに送信されます。XML Adapter は、このイベントをキューから受け入れ、XML データ・ハンドラーを起動して、イベントを XML 文書に変換します。この XML 文書は Web サーバーに POST されます。Web サーバーは、独自の XML で XML Adapter に応答します。

  – 2 番目の統合では、XML Adapter は XML 文書の URL をポーリングします。読み取ると、XML Adapter はその文書を XML データ・ハンドラーに送信します。データ・ハンドラーは応答イベントを戻します。この応答イベントは WebSphere MQ キューに送信され、WebSphere MQ Integrator Broker によって読み取られます。

- WebSphere InterChange Server 接続での XML サンプル・シナリオのインストール

  このシナリオ例では、2 方向のデータ交換を表す 2 つの統合が必要です。

  – 最初の統合では、まず「XML_REQUEST_Order」オブジェクトが PortConnector から「Port_To_XML」コラボレーション・オブジェクト経由で XML Adapter に送信されます。XML Adapter は XML データ・ハンドラーを起動して、要求ビジネス・オブジェクトを XML 文書に変換します。この XML 文書は Web サーバーに POST されます。Web サーバーは XML Adapter に応答し、XML Adapter はその応答 XML を応答ビジネス・オブジェクトに変換して、WebSphere ICS に戻します。

  – 2 番目の統合では、XML Adapter は XML 文書の URL をポーリングします。読み取ると、XML Adapter はその文書を XML データ・ハンドラーに送信します。この文書は応答ビジネス・オブジェクトに変換されて、WebSphere InterChange Server に送信されます。そして、2 つのコラボレーション「XML_To_Port_Customer」または「XML_To_Port_Manifest」のいずれかによって、イベントは Port Connector に送信されます。

WebSphere MQ Integrator Broker 接続での XML サンプル・シナリオのインストール

注: このサンプルでは、実際の WebSphere MQ Integrator Broker は使用しません。XML Adapter は、単に WebSphere MQ キューを読み書きするのみです。Portconnector を実行する Visual Test Connector を使用して WebSphere MQ Integrator Broker をシミュレートします。

サンプル・シナリオのインストールおよび検証手順を以下に示します。

- インストール前の注意事項および前提事項
- サンプル・シナリオのインストール
- サービス呼び出し要求シナリオの実行
- ポーリング・シナリオの実行
- 要約

インストール前の注意事項および前提事項

1. IBM WebSphere Business Integration Adapters をインストール済みで、その運用経験があること。
2. IBM WebSphere MQ をインストール済みで、その運用経験があること。
3. IBM WebSphere Business Integration Adapter for XML がインストール済みであること。
4. Java サーブレットを処理するように Web サーバーがセットアップされていること。
5. 本書の中で %CROSSWORLDS% は、現在インストールされている IBM WebSphere Business Integration Adapters を格納するフォルダーを指します。
6. すべての環境変数およびファイル分離文字は Windows NT/2000 の形式で記述されます。UNIX プラットフォームで実行する場合は、必要な変更を行ってください (例えば、%CROSSWORLDS%¥connectors であれば ${CROSSWORLDS}/connectors となります)。

サンプル・シナリオのインストール

1. **WebSphere MQ のセットアップ:**

   最初に、WebSphere MQ キュー・マネージャーを作成して始動します。チャネル・イニシエーターおよびリスナーも実行します。次に、以下の名前のキューを作成します。

   - ADMININQUEUE
   - ADMINOUTQUEUE
   - DELIVERYQUEUE
   - FAULTQUEUE
   - REQUESTQUEUE

2. **XML コネクターおよびポート・コネクターの CFG ファイルの構成:**

ファイル %CROSSWORLDS%¥connectors¥XML¥samples¥WebSphereMQIntegratorBroker¥XMLConnector.cfg を開きます。

現在のセットアップに応じて、ファイル内の以下のプロパティーを変更します。詳細は、「Guide to the IBM WebSphere Business Integration Adapter for XML」を参照してください。これらのプロパティーが表すパスまたはファイル名が存在しない場合は、そのパスまたはファイル名を作成する必要があります。

- QueueManager
- RepositoryDirectory
- ArchiveDirectory

ファイル %CROSSWORLDS%¥connectors¥XML¥samples¥WebSphereMQIntegratorBroker¥PortConnector.cfg を開きます。

現在のセットアップに応じて、ファイル内の以下のプロパティーを変更します。これらのプロパティーが表すパスまたはファイル名が存在しない場合は、そのパスまたはファイル名を作成する必要があります。

- QueueManager
- RepositoryDirectory

3. **XML コネクターの CFG ファイルの指定:**
   - ***NT/2000 の場合:***

     XML コネクターのショートカットのプロパティーを開きます。ターゲットの最後の引き数として、「-c」+ <XMLConnector.cfg ファイルの絶対パスおよびファイル名> を追加します。

     例えば、次のようになります。

     ```
     c%CROSSWORLDS%¥connectors¥XML¥samples¥WebSphereMQIntegratorBroker¥
     XMLConnector.cfg.
     ```

   - ***UNIX の場合:***

     ファイル ${CROSSWORLDS}/bin/connector_manager_XML を開きます。AGENTCONFIG_FILE プロパティーの値として、「-c」+ <XMLConnector.cfg ファイルの絶対パスおよびファイル名> を設定します。

     例えば、次のようになります。

     ```
     AGENTCONFIG_FILE=c${CROSSWORLDS}/connectors/XML/samples/
     WebSphereMQIntegratorBroker/XMLConnector.cfg.
     ```

4. **環境に応じたサーブレットの変更**

サーブレットについて、以下の変更を行う必要があります。

- PollXMLOrder.java の場合:

  ソース・ファイルの行 41 で、outFileName String の値を、サーブレットが着信 XML メッセージを記録する先となるローカル・システムのファイル名に変更します。ソース・ファイルの行 56 で、FileInputStream コンストラクターに渡される値は、提供された SamplePollingInput.xml ファイルのローカル・システムでのファイル名と正確に一致するようにしてください。

- MirrorXMLOrder.java の場合:

変更は不要です。

5. **XML Poll ビジネス・オブジェクトの構成:**

   Business Object Designer を始動します。「ファイル」 -> 「ファイルから開く」を選択し、「XML_POLL_Order.xsd」という名前のファイルを選択します。これにより、「XML_POLL_Order」というラベルの付いたビジネス・オブジェクトが開きます。このビジネス・オブジェクトの最初の属性の名前は「URL」です。URL 属性のデフォルト値を、XML Adapter が XML 応答 (PollXMLOrder.java など) を listen するロケーションに変更します。ファイル XML_POLL_Order.xsd を、変更したビジネス・オブジェクトに置き換えて、このビジネス・オブジェクトを保管します。

6. **Web サーバーの構成:**

   提供のサーブレットをコンパイルします。生成されたクラス・ファイルは、Web サーバーが選択して実行できるように、適切なディレクトリーに移す必要があります。さらに、サーブレットの登録に必要な追加の作業があれば、それを行います (必要な作業の詳細は、使用する Web サーバーによって異なります)。

サービス呼び出し要求シナリオの実行

1. **始動:**
   - XML Adapter を始動します。
   - WebServer を始動します。
   - Visual Test Connector の 1 つのインスタンスを始動します。
2. **ポート・コネクターのシミュレーション:**

   Visual Test Connector を使用して、「PortConnector」のプロファイルを定義します。Test Connector のメニューで、「ファイル」 -> 「プロファイルを開く (OPEN PROFILE)」を選択します。「追加」ボタンをクリックし、コネクター構成ファイルとして %CROSSWORLDS%¥connectors¥XML¥samples¥WebSphereMQIntegratorBroker¥PortConnector.cfg を指定します。この後に表示される 2 つのウィンドウで「OK」をクリックし、エージェントを接続します。

3. **テスト・データのロード:**

   「PortConnector」をシミュレートする Test Connector を使用して、メニューから「編集」 -> 「ビジネス・オブジェクトをロード」を選択します。次のファイルをロードしてください。

   %CROSSWORLDS%¥connectors¥XML¥samples¥WebSphereMQIntegratorBroker¥sampleOrderData.bo

4. **URL の設定:**

   Test Connector にロードしたテスト・データを開きます。URL 属性の値を、XML Adapter が XML 要求を POST するロケーションに変更します。

5. **テスト・データの送信:**

「PortConnector」をシミュレートする Test Connector を使用して、ロードされたテスト・ビジネス・オブジェクトをクリックします。メニューから「要求」 -> 「送信」を選択します。

6. **処理が正常に終了したことの確認:**

   処理が正常に終了したことを確認するには、XML Adapter がイベントを受信し、そのイベントを XML 文書に変換し、それを Web サーバーに POST し、応答を受信し、その応答を正常に解析したことを確認します。

ポーリング・シナリオの実行

1. **始動:**
   - XML Adapter
   - Web サーバー
   - Visual Test Connector の 1 つのインスタンス
2. **ポート・コネクターのシミュレーション:**

   Visual Test Connector を使用して、「PortConnector」のプロファイルを定義します。Test Connector のメニューで、「ファイル」 -> 「プロファイルを開く (OPEN PROFILE)」を選択します。「追加」ボタンをクリックし、コネクター構成ファイルとして %CROSSWORLDS%¥connectors¥XML¥samples¥WebSphereMQIntegratorBroker¥PortConnector.cfg を指定します。この後に表示される 2 つのウィンドウで「OK」をクリックし、エージェントを接続します。
3. **サンプル・データのポーリング:**
   - ***NT/2000 の場合:***

     PollFrequency はすでに key に設定されています。XML Adapter を始動したコマンド・ウィンドウで、文字「p」を入力して Enter キーを押します。XML Adapter を停止します。XMLConnector.cfg ファイルを開き、PollFrequency を 30000 (ポーリング/ミリ秒) 程度の値に変更して、ファイルを保管します。コネクターを再始動し、コネクターがポーリングするまで 30 秒待ちます。
   - ***UNIX の場合:***

     XML Adapter を停止します。XMLConnector.cfg ファイルを開き、PollFrequency を 30000 (ポーリング/ミリ秒) 程度の値に変更して、ファイルを保管します。コネクターを再始動し、コネクターがポーリングするまで 30 秒待ちます。
4. **ポート・コネクターを使用した要求の受け入れ:**

   XML Adapter は XML 文書を受信し、それをメッセージに変換して WebSphere MQ キューに格納します。Test Connector を使用して要求を受け入れ、正常応答で応答してください。
5. **処理が正常に終了したことの確認:**

処理が正常に終了したことを確認するには、Test Connector および Archive Directory の受け入れ済み要求のデータが、サンプルに提供されている SamplePollingInput.xml ファイルからのイベントと対応していることを確認します。

要約:

上記の手順をすべて正しく実行すれば、XML Adapter および XML データ・ハンドラーを使用して WebSphere MQ Integrator Broker と Web サーバーの間で XML 文書を交換するサンプル・シナリオを実施したことになります。

WebSphere InterChange Server 接続での XML サンプル・シナリオのインストール

注: このサンプルでは、ポーリングによって以下の 3 つのビジネス・オブジェクトが戻されます。

- XML_Order_Customer
- XML_Order_Manifest
- XML_Order_Receipt

これらのビジネス・オブジェクトのうち、2 つ (Customer および Manifest) にしかサブスクリプションを提供するコラボレーションがないため、3 番目のビジネス・オブジェクト (Receipt) は XML Adapter によって指定のロケーションにアーカイブされます。

サンプル・シナリオのインストールおよび検証手順を以下に示します。

- インストール前の注意事項および前提事項
- サンプル・シナリオのインストール
- サービス呼び出し要求シナリオの実行
- ポーリング・シナリオの実行
- 要約

インストール前の注意事項および前提事項

1. IBM WebSphere Business Integration Adapters をインストール済みで、その運用経験があること。
2. IBM WebSphere InterChange Server をインストール済みで、その運用経験があること。
3. IBM WebSphere Business Integration Adapter for XML がインストール済みであること。
4. Java サーブレットを処理するように Web サーバーがセットアップされていること。
5. 本書の中で %CROSSWORLDS% は、現在インストールされている IBM WebSphere Business Integration Adapters を格納するフォルダーを指します。
6. すべての環境変数およびファイル分離文字は Windows NT/2000 の形式で記述されます。UNIX プラットフォームで実行する場合は、必要な変更を行ってくださ

い (例えば、%CROSSWORLDS%¥connectors であれば ${CROSSWORLDS}/connectors となります)。

サンプル・シナリオのインストール

1. **ビジネス・オブジェクトをリポジトリーにロード:**

   IBM WebSphere ICS Interchange Server を始動し、WebSphere Business Integration System Manager を使用して、Business Object Designer の「ファイル」メニューの「ファイルから開く」メニュー項目を選択します。%CROSSWORLDS%¥connectors¥XML¥Samples¥WebSphereICS フォルダーにある、「Sample_XML_Order_Objects.in」というラベルの付いたリポジトリー・ファイルをロードします。ビジネス・オブジェクトがロードされたことを確認してください。全部で 12 個あります。

2. **コネクターをリポジトリーにロード:**

   WebSphere Business Integration System Manager を使用して、Connector Designer の「ファイル」メニューの「ファイルから開く」メニュー項目を選択します。%CROSSWORLDS%¥connectors¥XML¥Samples¥WebSphereICS フォルダーにある、「Sample_XML_Order_Connectors.in」というラベルの付いたリポジトリー・ファイルをロードします。XMLConnector 定義および PortConnector 定義がロードされたことを確認してください。

3. **XML コネクターの構成:**

   WebSphere Business Integration System Manager を使用して、XML コネクター定義をダブルクリックします。これにより Connector Designer が起動します。ファイル構造に応じて、以下のアプリケーション構成プロパティー値を変更してください。このパスまたはファイル (あるいはその両方) が存在しない場合は、作成する必要があります。

   - ArchiveDirectory

4. **コラボレーション・テンプレートおよびオブジェクトをリポジトリーにロード**

   WebSphere Business Integration System Manager を使用して、「ファイル」メニューの「ファイルから開く」メニュー項目を選択します。%CROSSWORLDS%¥connectors¥XML¥samples¥WebSphereICS フォルダーにある、「Sample_XML_Order_Collaborations.in」というラベルの付いたリポジトリー・ファイルをロードします。3 つのテンプレート定義および 3 つのコラボレーション・オブジェクトがロードされたことを確認してください。

5. **コラボレーション・テンプレートのコンパイル:**

   WebSphere Business Integration System Manager を使用して、「*コラボレーション・テンプレート (Collaboration Templates)*」 というラベルの付いたフォルダーを右マウス・ボタン・クリックし、ドロップダウン・リストから**「すべてコンパイル」**を選択します。

6. **環境に応じたサーブレットの変更**

   サーブレットについて、以下の変更を行う必要があります。

   - PollXMLOrder.java の場合:

ソース・ファイルの行 41 で、outFileName String の値を、サーブレットが着信 XML メッセージを記録する先となるローカル・システムのファイル名に変更します。ソース・ファイルの行 56 で、FileInputStream コンストラクターに渡される値は、提供された SamplePollingInput.xml ファイルのローカル・システムでのファイル名と正確に一致するようにしてください。

- MirrorXMLOrder.java の場合:

  変更は不要です。

7. **XML Poll ビジネス・オブジェクトの構成:**

   WebSphere Business Integration System Manager から、「XML_POLL_Order」というラベルの付いたビジネス・オブジェクトを開きます。このビジネス・オブジェクトの最初の属性の名前は「URL」です。URL 属性のデフォルト値を、XML Adapter が XML 応答 (PollXMLOrder.java など) を listen するロケーションに変更します。ビジネス・オブジェクトをサーバーに保管します。

8. **Web サーバーの構成:**

   提供のサーブレットをコンパイルします。生成されたクラス・ファイルは、Web サーバーが選択して実行できるように、適切なディレクトリーに移す必要があります。さらに、サーブレットの登録に必要な追加の作業があれば、それを行います (必要な作業の詳細は、使用する Web サーバーによって異なります)。

9. **WebSphere InterChange Server 再始動:**

   すべての変更を有効にするため、Interchange Server をリブートします。WebSphere Business Integration System Manager のシステム表示を使用して、すべてのコラボレーション・オブジェクトおよびコネクター・コントローラーが正常であることを確認してください。

サービス呼び出し要求シナリオの実行

1. **始動:**
   - WebSphere Interchange Server (まだ稼働していない場合)
   - XML Adapter
   - Web サーバー
   - Visual Test Connector の 1 つのインスタンス
2. **ポート・コネクターのシミュレーション:**

   Test Connector を使用して、「PortConnector」のプロファイルを定義します。エージェントのシミュレーションを開始するため、Test Connector のメニューから「ファイル」 -> 「エージェントの接続 (CONNECT AGENT)」を選択します。

3. **テスト・データのロード:**

   「PortConnector」をシミュレートする Test Connector を使用して、メニューから「編集」 -> 「ビジネス・オブジェクトをロード」を選択します。次のファイルをロードしてください。

   ```
   %CROSSWORLDS%¥connectors¥XML¥samples¥WebSphereICS¥
   sampleOrderData.bo
   ```

4. **URL の設定:**

Test Connector にロードしたテスト・データを開きます。URL 属性の値を、XML Adapter が XML 要求を POST するロケーションに変更します。

5. **テスト・データの送信:**

   「PortConnector」をシミュレートする Test Connector を使用して、ロードされたテスト・ビジネス・オブジェクトをクリックします。メニューから「要求」 -> 「送信」を選択します。

6. **処理が正常に終了したことの確認:**

   処理が正常に終了したことを確認するには、XML Adapter がイベントを受信し、そのビジネス・オブジェクトを XML 文書に変換し、それを Web サーバーに POST し、応答を受信し、その応答を解析してコラボレーションに送信したことを確認します。

ポーリング・シナリオの実行

1. **始動**
   - WebSphere Interchange Server (まだ稼働していない場合)
   - XML Adapter
   - Web サーバー
   - Visual Test Connector の 1 つのインスタンスを始動します。
2. **ポート・コネクターのシミュレーション:**

   Test Connector を使用して、「PortConnector」のプロファイルを定義します。エージェントのシミュレーションを開始するため、Test Connector のメニューから「ファイル」 -> 「エージェントの接続 (CONNECT AGENT)」を選択します。

3. **サンプル・データのポーリング:**
   - ***NT/2000 の場合:***

     PollFrequency はすでに key に設定されています。XML Adapter を始動したコマンド・ウィンドウで、文字「p」を入力して Enter キーを押します。

   - ***UNIX の場合:***

     XML Adapter を停止します。XML Adapter Controller を開き、PollFrequency を 30000 (ポーリング/ミリ秒) 程度の値に変更します。コネクターを再始動し、コネクターがポーリングするまで 30 秒待ちます。

4. **ポート・コネクターを使用した要求の受け入れ:**

   XMLConnector は XML 文書を受信し、それを CrossWorlds ビジネス・オブジェクトに変換して、WebSphere InterChange Server に渡します。WebSphere ICS は、このイベントのサブスクリプションを持つ 2 種類のコラボレーションにイベントを渡します。コラボレーションは PortConnector にイベントを渡します。Test Connector を使用して要求を受け入れ、両方のイベントに正常応答で応答します。

5. **処理が正常に終了したことの確認:**

処理が正常に終了したことを確認するには、Test Connector および Archive Directory の受け入れ済み要求のデータが、サンプルに提供されている SamplePollingInput.xml ファイルからのイベントと対応していることを確認します。

要約:

上記の手順をすべて正しく実行すれば、XML Adapter および XML データ・ハンドラーを使用して IBM WebSphere ICS と Web サーバーの間で XML 文書を交換するサンプル・シナリオを実施したことになります。

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

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

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

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

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

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

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

著作権使用許諾

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

プログラミング・インターフェース情報

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

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

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

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

商標

以下は、IBM Corporation の商標です。

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

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

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

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

WebSphere Business Integration Adapter Framework V2.4.0

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ハ行]

[マ行]

[ヤ行]

A

B

C

D

G

H

I

J

K

M

P

R

S

V

X

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