

**IBM WebSphere Business Integration
Adapters**



Adapter for WebSphere Commerce ユーザーズ・ガイド

Adapter バージョン 2.5.0

**IBM WebSphere Business Integration
Adapters**



Adapter for WebSphere Commerce ユーザーズ・ガイド

Adapter バージョン 2.5.0

お願い

本書および本書で紹介する製品をご使用になる前に、103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 WebSphere Commerce バージョン 2.5.0 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原 典： IBM WebSphere Business Integration Adapters
Adapter for WebSphere Commerce User Guide
Adapter Version 2.5.0

発 行： 日本アイ・ビー・エム株式会社

担 当： ナショナル・ランゲージ・サポート

第1刷 2004.1

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

© Copyright International Business Machines Corporation 1997, 2003. All rights reserved.

© Copyright IBM Japan 2004

まえがき

IBM[®] WebSphere[®] 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、およびレガシー・システムとメインフレーム・システムに、統合接続性を提供します。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 WebSphere Commerce のインストール、構成、トラブルシューティング、およびビジネス・オブジェクト開発について説明します。

対象読者

本書は、お客様のサイトでコネクターを使用するコンサルタント、開発者、およびシステム管理者を対象とします。

本書の前提条件

本書では、IBM WebSphere InterChange Server、ビジネス・オブジェクトとコラボレーションの開発、WebSphere Commerce アプリケーション、および WebSphere MQ について十分な知識と経験を持っている必要があります。

関連資料

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。

- アダプターの一般情報、WebSphere Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) におけるアダプターの使用、および WebSphere Application Server におけるアダプターの使用については、次のアドレスの IBM WebSphere Business Integration Adapters InfoCenter を参照してください。
<http://www.ibm.com/websphere/integration/wbiadapters/infocenter>
- WebSphere InterChange Server におけるアダプターの使用については、次のアドレスの IBM WebSphere InterChange Server InfoCenters を参照してください。
<http://www.ibm.com/websphere/integration/wicsserver/infocenter>
<http://www.ibm.com/websphere/integration/wbicollaborations/infocenter>
- WebSphere Message Brokers の詳細については、以下を参照してください。
<http://www.ibm.com/software/integration/mqfamily/library/manualsa/>
- WebSphere Application Server の詳細については、以下を参照してください。
<http://www.ibm.com/software/webservers/appserv/library.html>

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

表記上の規則

本書では、以下の規則を使用します。

Courier フォント	コマンド名、ファイル名、入力する情報、システムが画面に出力する情報などのリテラル値を示します。
太字	初出語を示します。
イタリック	変数名または相互参照を示します。
青い文字	オンラインで表示したとき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。青い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飛ぶことができます。
{ }	構文の説明で、複数のオプションが中括弧で囲まれている場合、その中の 1 つのオプションのみを選択することが必要です。
[]	構文の説明で、大括弧で囲まれているパラメーターはオプションナルです。
...	構文の説明で、省略符号は、直前のパラメーターの繰り返しを示します。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 つの名前の各エレメントを個々に判別できるようにするために、不等号括弧で囲みます。例： <code><server_name><connector_name>tmp.log</code>
/ および ¥	本書では、ディレクトリー・パスの区切り記号として円記号 (¥) 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。すべての製品のパス名は、ご使用のシステムで製品がインストールされているディレクトリーを基準とした相対パス名です。
----UNIX/Windows----	UNIX または Windows のいずれかが先頭に明記されている段落は、オペレーティング・システムの違いを表す注意事項を示します。
%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 環境ではこれを \$text で表します。すなわち、text UNIX 環境変数の値を示します。
ProductDir	IBM WebSphere Business Integration Adapters 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。

目次

まえがき	iii
対象読者	iii
本書の前提条件	iii
関連資料	iii
表記上の規則	iv
本リリースの新機能	vii
リリース 2.5.0 の新機能	vii
リリース 2.3.x の新機能	vii
第 1 章 概要	1
アダプターのアーキテクチャー	2
アプリケーションとアダプター間の通信	4
イベント処理	6
保証付きイベント・デリバリー	8
ビジネス・オブジェクト要求	8
動詞の処理	9
共通の構成タスク	12
第 2 章 コネクターのインストールおよび構成	17
アダプター環境	17
前提条件となる作業	18
アダプターおよび関連ファイルのインストール	30
アダプターの構成	30
保証付きイベント・デリバリー機能の使用可能化	35
キューの URI (Uniform Resource Identifiers)	40
メタオブジェクト属性の構成	42
アダプターの複数インスタンスの作成	51
始動ファイルの構成	52
コネクターの開始	53
コネクターの停止	54
第 3 章 ビジネス・オブジェクトの処理	55
ビジネス・オブジェクトの構造	55
エラー処理	56
トレース	57
第 4 章 トラブルシューティング	59
始動時の問題	59
イベント処理	60
付録 A. コネクターの標準構成プロパティー	61
新規プロパティーと削除されたプロパティー	61
標準コネクター・プロパティーの構成	61
標準プロパティーの要約	63
標準構成プロパティー	67
付録 B. Connector Configurator	79
Connector Configurator の概要	79
Connector Configurator の始動	80

System Manager からの Configurator の実行	81
コネクタ固有のプロパティ・テンプレートの作成	81
新規構成ファイルの作成	84
既存ファイルの使用	85
構成ファイルの完成	86
構成ファイル・プロパティの設定	87
構成ファイルの保管	94
構成ファイルの変更	95
構成の完了	95
グローバル化環境における Connector Configurator の使用	95
付録 C. InFashion サンプルの実行	97
IBM WebSphere Business Integration Adapter for WebSphere Commerce のインストール	97
前提条件	98
サンプル・シナリオのインストール	98
シナリオの実行	101
アダプターのトラブルシューティング	102
特記事項	103
プログラミング・インターフェース情報	104
商標	105

本リリースの新機能

リリース 2.5.0 の新機能

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

バージョン 2.5 から、Adapter for WebSphere Commerce は Microsoft Windows NT 上ではサポートされなくなりました。

リリース 2.3.x の新機能

2003 年 3 月に更新されました。「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。

データ・ハンドラーを入力キューと関連付けることができるようになりました。詳細については、45 ページの『データ・ハンドラーから InputQueues へのマッピング』を参照してください。

保証付きイベント・デリバリー機能が拡張されました。詳細については、35 ページの『保証付きイベント・デリバリー機能の使用可能化』を参照してください。

第 1 章 概要

この章では、IBM WebSphere Business Integration Adapter for WebSphere Commerce について説明します。

このアダプターを使用すると、IBM WebSphere Integration Server アダプターと WebSphere Commerce Business Edition バージョン 5.4 (修正パッケージ 2 または 5.5) 間でメッセージ交換を行えるようになります。この章は、次のトピックから構成されます。

- 2 ページの『アダプターのアーキテクチャー』
- 4 ページの『アプリケーションとアダプター間の通信』
- 6 ページの『イベント処理』
- 8 ページの『保証付きイベント・デリバリー』
- 8 ページの『ビジネス・オブジェクト要求』
- 9 ページの『動詞の処理』
- 12 ページの『共通の構成タスク』

WebSphere Commerce ソフトウェアは、さまざまなコマース統合の役割を果たすことができる、フレキシブルなプラットフォームです。Adapter for WebSphere Commerce は、統合ブローカーを使用して、WebSphere Commerce と、すでに適切なアダプターがインストールされているその他のエンタープライズ情報システム・アプリケーションとの間のビジネス・データ交換を統合するソリューションで使用することができます。

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構成されています。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合わせたコードが格納されています。アダプター・フレームワークは、コードがすべてのアダプターに共通なので、統合ブローカーとアプリケーション固有のコンポーネントとの仲介役の機能を果たします。アダプター・フレームワークが、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間で提供するサービスは、以下のとおりです。

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

本書では、アダプター・フレームワークとコネクターについて説明します。これらのコンポーネントのいずれも、アダプターと呼ばれます。統合ブローカーとアダプ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*」を参照してください。

注: すべての WebSphere Business Integration アダプターは、統合ブローカーと連携して動作します。Adapter for WebSphere Commerce は、InterChange Server (ICS) 統合ブローカーと連動します。詳細については、「*テクニカル入門 (IBM WebSphere InterChange Server)*」を参照してください。

アダプターのアーキテクチャー

アダプターは、メタデータ主導型です。アダプターは、JavaTM Message Service (JMS) の MQ インプリメンテーション、つまり、エンタープライズ・メッセージング・システムにアクセスするための API を使用します。

アダプターは、WebSphere MQ のキューを使用して、WebSphere Commerce から ICS、および ICS から WebSphere Commerce への非同期データ交換を実現します。データは、WebSphere Commerce のキューと ICS との間で XML メッセージ形式でやり取りされます。ICS コラボレーション・オブジェクトで処理できるビジネス・オブジェクトへのデータ変換には、XML データ・ハンドラーを使用します。

同期交換でのアダプターの使い方については、3 ページの『要求と応答の同期対話』を参照してください。

WebSphere Commerce から統合ブローカーへの非同期メッセージ

注文が WebSphere Commerce に入ると、OrderCreate メッセージが XML 形式で生成され、以下の図に示すように WebSphere MQ 出力キューに入ります。この図では、WebSphere Commerce と統合ブローカーは、異なるキュー・マネージャーを使用する別々のマシンにインストールされているので、WebSphere Commerce からの出力用のリモート・キュー定義を用意し、統合ブローカーから見るとローカルにある入力キューに接続することが必要であるものとします。WebSphere Commerce と統合ブローカーが同じマシンにインストールされている場合は、1 つのキューで、WebSphere Commerce からの出力キューと統合ブローカー用の入力キューの両方の役割を果たすことができます。

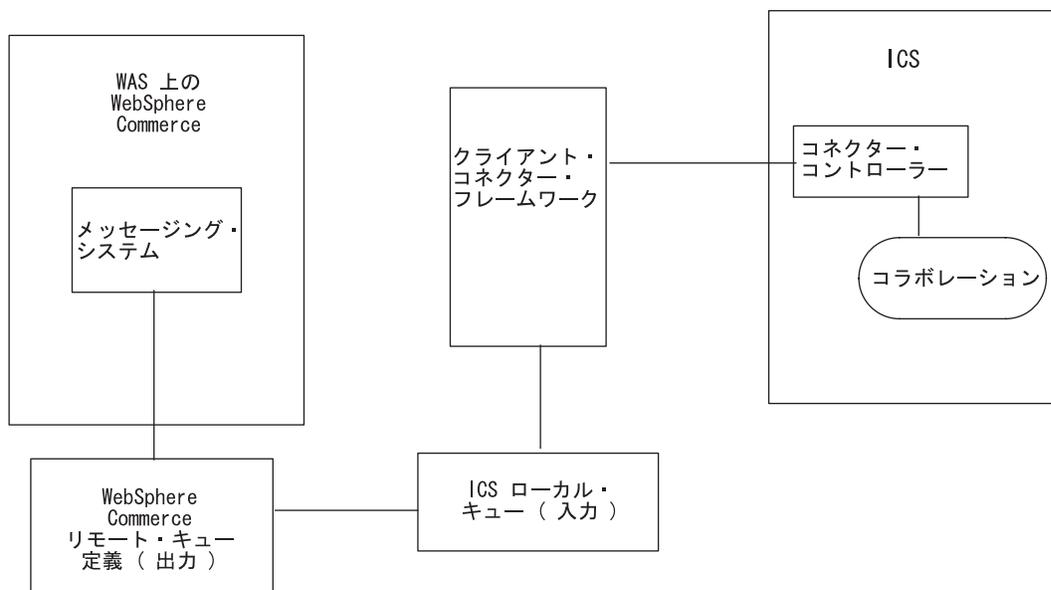


図 1. アダプターのアーキテクチャー

WebSphere Commerce でのデータ・イベントを検出するため、アダプターは WebSphere Commerce 出力キューをポーリングして、新しい XML メッセージを検索します。新しいメッセージが見つかったら、アダプターはそれを入力キューに渡し、データ・ハンドラーを呼び出して、そのメッセージを、WebSphere Commerce から出力されるデータの構造に固有のビジネス・オブジェクトに変換してから、そのビジネス・オブジェクトを ICS 内のコネクタに渡します。コネクタは、マップを呼び出して、WebSphere Commerce 固有のビジネス・オブジェクトから汎用のビジネス・オブジェクトを生成し、その汎用のビジネス・オブジェクトを 1 つ以上のコラボレーション・オブジェクトに引き渡します。コラボレーション・オブジェクトによるビジネス・オブジェクトの処理が終了すると、汎用のビジネス・オブジェクトはアプリケーション固有のビジネス・オブジェクトにマップされて、バックエンド・アプリケーションに合わせて構成されているアダプター (SAP 用の WebSphere Business Integration アダプターなど) に引き渡されます。

ICS から WebSphere Commerce への非同期メッセージ

反対に、ICS から WebSphere への送信では、Adapter for WebSphere Commerce は、コラボレーションからビジネス・オブジェクトを受け取り、データ・ハンドラーを使用してそれらを XML 形式のメッセージに変換してから、そのメッセージを WebSphere Commerce WebSphere MQ キューに引き渡します。

要求と応答の同期対話

要求と応答の同期対話では、以下のトピックで説明するように、WebSphere Commerce アプリケーションへの追加やカスタマイズが必要となります。

WebSphere Commerce から InterChange Server への要求

WebSphere Commerce 拡張パック (

<http://www-3.ibm.com/software/webservers/commerce/epacks/v54/> で入手可能) を追加すれば、アダプターを使用して、WebSphere Commerce メッセージング・システムから InterChange Server などの外部システムへの要求と応答の対話で同期メッセージ・フローを設定することができます。この方法の詳細については、WebSphere Commerce 5.4 および IBM WebSphere Business Integration システムの統合資料を参照してください。

InterChange Server から WebSphere Commerce への要求

注: この方法を行う場合は、ICS からビジネス・オブジェクトを受け取ったときに WebSphere Commerce で実行するコマンドをカスタマイズする必要があります。このコマンドは、メッセージから「ReplyTo」キューを取得し、ResponseTimeout 間隔内に応答をそのキューに入れる、というものです。WebSphere Commerce でのコマンドの作成とカスタマイズについては、「*WebSphere Commerce プログラマーズ・ガイド 5.4*」を参照してください。

WebSphere Commerce 拡張パックを使用せずに、Replyto キューを使って、WebSphere Commerce と ICS 間の同期交換のシミュレーションを設定することもできます。

この方法の詳細については、9 ページの『同期の引き渡し』を参照してください。

イベント通知

WebSphere Commerce アプリケーションで発生したデータ・イベントの通知は、アダプターのポーリング機構を介して実現されます。アダプターは、複数の入力キューをポーリングし、ラウンドロビン方法で各キューをポーリングしながら、それぞれのキューから指定された数のメッセージを取得します。ポーリング時に取得したメッセージごとに、アダプターは動的子メタオブジェクトを追加します (ビジネス・オブジェクトで指定されている場合)。子メタオブジェクトの値を使用して、アダプターは、メッセージのフォーマットの属性と、メッセージの取得元の入力キューの名前の属性を生成することができます。

入力キューからメッセージを取得すると、アダプターは、メッセージ・ヘッダーの `FORMAT` フィールドに関連付けられたビジネス・オブジェクト名を探します。次に、メッセージ本体が、該当するビジネス・オブジェクトの新しいインスタンスと共に、データ・ハンドラーに渡されます。フォーマットに関連付けられたビジネス・オブジェクト名が見つからない場合は、メッセージ本体だけがデータ・ハンドラーに渡されます。メッセージの内容からビジネス・オブジェクトが正常に生成された場合、アダプターは、それがサブスクライブされているかどうかを確認してから、InterChange Server に引き渡します。

ビジネス・オブジェクトと WebSphere MQ のメッセージ・ヘッダー

メッセージを処理する場合に使用するビジネス・オブジェクトと動詞のタイプは、WebSphere MQ メッセージ・ヘッダーに収められている `FORMAT` フィールドに基づきます。アダプターは、メタオブジェクトの項目を使用して、ビジネス・オブジェクトの名前と動詞を判断します。WebSphere MQ メッセージ・ヘッダーの `FORMAT` フィールド・テキストと関連付けるビジネス・オブジェクト名と動詞を格納する、メタオブジェクトを作成します。

必要に応じて、アダプターに渡すビジネス・オブジェクトに子として追加する動的メタオブジェクトを作成することもできます。子メタオブジェクトの値は、アダプター全体に対して指定される静的メタオブジェクトで指定されている値をオーバーライドします。子メタオブジェクトが定義されていない、または必要な変換プロパティが定義されていない場合、アダプターは、デフォルトで、静的メタオブジェクトの値を調べます。単一の静的アダプター・メタオブジェクトの代わりに、または補足として、1 つ以上の動的子メタオブジェクトを指定することができます。

アプリケーションとアダプター間の通信

アダプターは、Java Message Service (JMS) の IBM の WebSphere MQ インプリメンテーションを利用します。JMS は、エンタープライズ・メッセージング・システムにアクセスするためのオープン・スタンダードの API です。JMS の目的は、ビジネス・アプリケーションが、ビジネス・データとイベントを非同期で送受信できるようにすることです。

メッセージ要求

図 2 に、メッセージ要求通信を示します。doVerbFor() メソッドがコラボレーションからビジネス・オブジェクトを受け取ると、アダプターはそのビジネス・オブジ

エクトをデータ・ハンドラーに渡します。データ・ハンドラーは、ビジネス・オブジェクトを XML テキストに変換し、アダプターはメッセージとしてそのテキストをキューに発行します。そこで、JMS レイヤーが、キュー・セッションを開いてメッセージを転送するための適切な呼び出しを行います。

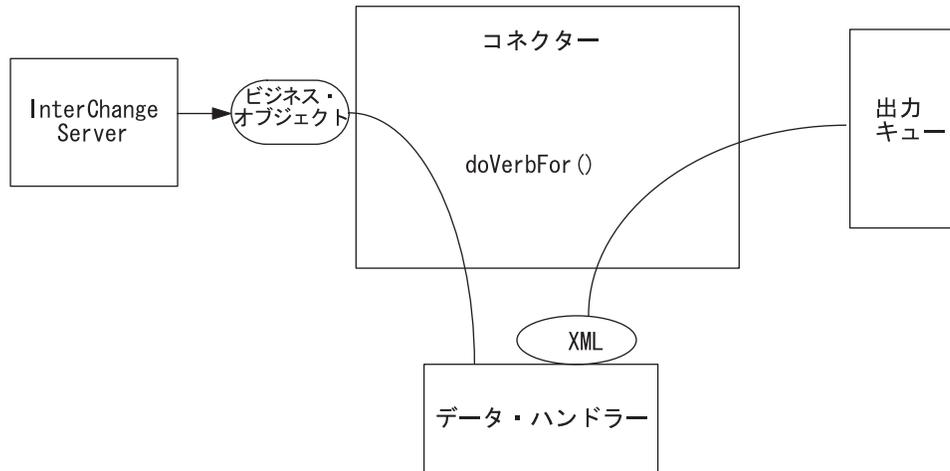


図2. アプリケーションとアダプター間の通信方式: メッセージ要求

メッセージの戻り

図3 に、メッセージが戻る方向を示します。pollForEvents() メソッドが、入力キューから次に該当するメッセージを取得します。メッセージは、処理中キューに入り、処理が完了するまでそこに置かれます。静的または動的いずれかのメタオブジェクトを使用して、アダプターは、まずメッセージ・タイプがサポートされているかどうかを判断します。サポートされていれば、構成されているデータ・ハンドラーにそのメッセージを渡し、そのデータ・ハンドラーがメッセージをビジネス・オブジェクトに変換します。設定されている動詞は、メッセージ・タイプについて設定されている変換プロパティを反映します。その後、アダプターはビジネス・オブジェクトがコラボレーションでサブスクライブされているかどうかを判断します。サブスクライブされている場合は、getAppEvents() メソッドがビジネス・オブジェクトを InterChange Server に引き渡して、そのメッセージは処理中キューから除去されます。

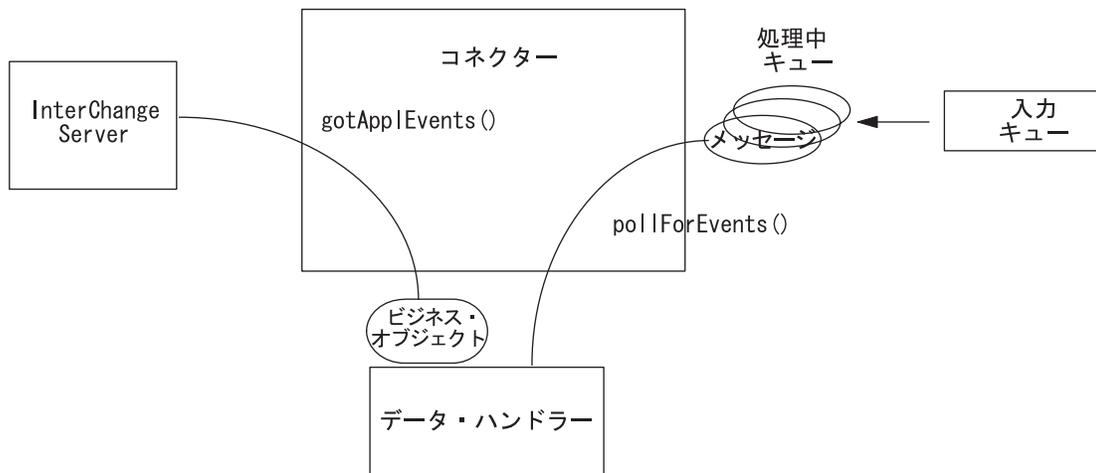


図3. アプリケーションとアダプター間の通信方式: メッセージの戻り

イベント処理

イベント通知では、アダプターは、WebSphere Commerce がキューに書き込んだイベントを検出します。

検索

アダプターは、pollForEvents() メソッドを使用して、一定の間隔でキューをポーリングしてメッセージを検索します。メッセージが見つかったら、アダプターはキューからそのメッセージを取得し、調べて、フォーマットを判断します。フォーマットがアダプターの静的メタオブジェクトで定義されている場合、アダプターはメッセージ本体と、そのフォーマットに関連付けられているビジネス・オブジェクトの新しいインスタンスの両方を、構成されているデータ・ハンドラーに渡します。データ・ハンドラーでは、ビジネス・オブジェクトが生成されて、動詞が指定されるはずですが、フォーマットが静的メタオブジェクトで定義されていない場合は、アダプターはメッセージ本体だけをデータ・ハンドラーに渡します。データ・ハンドラーでは、そのメッセージに対する適正なビジネス・オブジェクトが判別され、作成され、生成されるはずですが、イベント障害のシナリオについては、56 ページの『エラー処理』を参照してください。

アダプターは、メッセージを処理するときに、まず入力キューに対するトランザクション・セッションを開きます。このトランザクション方法では、わずかですが、アダプターがビジネス・オブジェクトを正常に実行依頼しても、キューにトランザクションをコミットできないと、ビジネス・オブジェクトがコラボレーションに 2 回引き渡される可能性があります。この問題を回避するため、アダプターはすべてのメッセージを処理中キューに移動します。処理が完了するまで、メッセージはそこに保持されます。アダプターが処理中に予想外のシャットダウンを起こしても、メッセージは処理中キューに残り、オリジナルの入力ファイルに復元されることはありません。

注: WebSphere MQ を使ったトランザクション・セッションでは、キュー上の要求アクションがすべて実行され、コミットされた後で、キューからイベントを除去する必要があります。したがって、アダプターは、キューからメッセージを取得するとき、次の 3 つの条件のいずれかが発生するまで、その取得をコミットしません。1) メッセージがビジネス・オブジェクトに変換される 2) ビジネス・オブジェクトが `getApplEvents()` メソッドによって InterChange Server に引き渡される 3) 戻り値を受け取る

リカバリー

初期化時に、アダプターは、処理中キュー内にコネクターのシャットダウンなどが原因で処理が完全に終了していないメッセージがないかどうかを確認します。コネクタ構成プロパティ `InDoubtEvents` で、このようなメッセージのリカバリーを処理する 4 つのオプション (「始動時に異常終了 (`fail on startup`)」、「再処理 (`reprocess`)」、「無視 (`ignore`)」、または「ログ・エラー (`log error`)」) のいずれかを指定することができます。

始動時に異常終了 (`fail on startup`)

「始動時に異常終了 (`fail on startup`)」オプションを指定した場合、初期化時に処理中キュー内にメッセージが見つかり、アダプターはエラーをログに記録して、即時にシャットダウンします。メッセージを調べて、これらのメッセージを完全に削除するか、別のキューに移動するか、どちらか適切なアクションを取ることは、ユーザーまたはシステム管理者の責任となります。

再処理 (`reprocess`)

「再処理 (`reprocessing`)」オプションを指定した場合、初期化時に処理中キュー内にメッセージが見つかり、アダプターは次のポーリング時にまずこれらのメッセージを処理します。処理中キュー内のメッセージの処理がすべて終了したら、アダプターは入力キューのメッセージの処理を開始します。

無視 (`ignore`)

「無視 (`ignore`)」オプションを指定した場合、初期化時に処理中キュー内にメッセージが見つかり、アダプターはそれらを無視します。ただし、シャットダウンはしません。

ログ・エラー (`log error`)

「ログ・エラー (`log error`)」オプションを指定した場合、初期化時に処理中キュー内にメッセージが見つかり、アダプターはエラーをログに記録します。ただし、シャットダウンはしません。

アーカイブ

コネクタ・プロパティ `ArchiveQueue` で、有効なキューが指定されている場合、アダプターは、正常に処理されたすべてのメッセージのコピーをアーカイブ・キューに入れます。`ArchiveQueue` が定義されていない場合は、メッセージは処理後に廃棄されます。アンサブスクライブされたメッセージやエラーが発生したメッセージのアーカイブについては、56 ページの『エラー処理』を参照してください。

注: JMS 規則により、取得したメッセージを即時に別のキューに発行することはできません。メッセージのアーカイブと再引き渡しを実現するためには、アダプ

ターはまずオリジナル・メッセージの本体とヘッダー (該当する場合) を複製した 2 つ目のメッセージを作ります。WebSphere Commerce メッセージング・サービスとの競合を避けるため、JMS に必要なフィールドだけが複製されます。したがって、フォーマット・フィールドが、アーカイブまたは再引き渡しされるメッセージについてコピーされる唯一の追加メッセージ・プロパティです。

保証付きイベント・デリバリー

保証付きイベント・デリバリー機能により、コネクタ・フレームワークは、コネクタのイベント・ストアと、JMS イベント・ストアおよび宛先の JMS キューの間で、イベントを失ったり 2 回送ったりせずに、確実に送信することができます。JMS 対応にするためには、`connectorDeliveryTransport` 標準プロパティに JMS を設定する必要があります。このように構成されたコネクタは、JMS トランスポートを使用し、コネクタと統合ブローカーとの間の以降の通信は、すべてこのトランスポートを介して行われます。JMS トランスポートにより、メッセージは最終的に宛先に確実に配送されます。JMS トランスポートの役割は、トランザクション・キュー・セッションが開始されると、コミットが発行されるまでメッセージがキャッシュされるようにすることです。障害が発生するかまたはロールバックが発行されると、メッセージは破棄されます。

注: 保証付きイベント・デリバリー機能を使用しないと、コネクタがイベントをパブリッシュして (コネクタが `pollForEvents()` メソッド内の `gotApplEvent()` メソッドを呼び出して) から、イベント・レコードを削除してイベント・ストアを更新する (または「イベント通知済み」状況に更新する) までの間に、障害の可能性を示す短い間が空きます。このすき間で障害が発生すると、イベントは送信されますが、イベント・レコードはイベント・ストアで「進行中」状況のままになっています。コネクタは再始動時に、このイベント・ストアに残されたイベント・レコードを検出して送信するので、イベントが 2 回送信されることになります。

JMS イベント・ストアの有無にかかわらず、JMS 対応コネクタに保証付きイベント・デリバリー機能を構成することができます。保証付きイベント・デリバリー用にコネクタを構成する方法については、35 ページの『保証付きイベント・デリバリー機能の使用可能化』を参照してください。

コネクタ・フレームワークがビジネス・オブジェクトを ICS 統合ブローカーに配送できない場合、オブジェクトは (`UnsubscribedQueue` と `ErrorQueue` ではなく) `FaultQueue` に配置されて、状況表示と問題の説明を生成します。`FaultQueue` メッセージは MQRFH2 フォーマットで書き込まれます。

ビジネス・オブジェクト要求

InterChange Server がビジネス・オブジェクトを `doVerbFor()` メソッドに送信すると、ビジネス・オブジェクト要求が処理されます。構成されているデータ・ハンドラーを使用して、アダプターはビジネス・オブジェクトを WebSphere MQ メッセージに変換し、発行します。データ・ハンドラーの要件を除き、処理対象のビジネス・オブジェクトのタイプに関する要件はありません。

動詞の処理

アダプターは、各ビジネス・オブジェクトに対する動詞に基づいて、コラボレーションから渡されたビジネス・オブジェクトを処理します。アダプターは、ビジネス・オブジェクト・ハンドラーと `doForVerb()` メソッドを使用して、アダプターがサポートするビジネス・オブジェクトを処理します。アダプターがサポートするビジネス・オブジェクトの動詞は、以下のとおりです。

- Create
- Update
- Delete
- Retrieve
- Exists
- Retrieve by Content

注: Create、Update、Delete の動詞を使用するビジネス・オブジェクトは、非同期、同期のどちらでも発行できます。デフォルト・モードは非同期です。アダプターは、Retrieve、Exists、Retrieve by Content の動詞を使用するビジネス・オブジェクトについては、非同期転送をサポートしていません。したがって、Retrieve、Exists、Retrieve by Content の動詞の場合のデフォルト・モードは、同期です。

Create、Update、Delete

Create、Update、Delete の動詞を使用するビジネス・オブジェクトの処理は、そのオブジェクトの発行が非同期か同期かによって異なります。

非同期の引き渡し

Create、Update、Delete の動詞を使用するビジネス・オブジェクトのデフォルトの転送モードは、非同期です。メッセージは、データ・ハンドラーを使用してビジネス・オブジェクトから作成され、出力キューに書き込まれます。メッセージが引き渡されると、アダプターは `BON_SUCCESS` を戻します。引き渡されなかった場合は、`BON_FAIL` を戻します。

注: アダプターには、メッセージを受け取ったかどうかや、アクションが実行されたかどうかを検証する方法はありません。

同期の引き渡し

注: この方法を行うには、ICS からビジネス・オブジェクトを受け取ったときに WebSphere Commerce で実行するコマンドをカスタマイズする必要があります。このコマンドは、メッセージから「ReplyTo」キューを取得し、ResponseTimeout 間隔内に応答をそのキューに入れる、というものです。WebSphere Commerce でのコマンドの作成とカスタマイズについては、「*WebSphere Commerce プログラマーズ・ガイド 5.4*」を参照してください。

コネクター・プロパティーで `replyToQueue` が定義されていて、`ResponseTimeout` がビジネス・オブジェクトの変換プロパティーに存在する場合、アダプターは同期モードで要求を発行します。次に、アダプターは、応答を待機し、WebSphere Commerce によって適切なアクション行われたことを検証します。

アダプターは、まず、表 1 に示すようなヘッダー付きのメッセージを発行します。

表 1. 要求メッセージの記述子ヘッダー (MQMD)

フィールド	説明	値
Format	フォーマットの名前。	変換プロパティーで定義されている出力フォーマット。IBM 要件に従い 8 文字に切り捨てられます (例: MQSTR)。
MessageType	メッセージのタイプ。	MQMT_DATAGRAM*
Report	要求されたレポート・メッセージのオプション。	<p>応答メッセージが必要な場合は、このフィールドに以下の値を指定します。</p> <p>MQRO_PAN*: 処理が成功した場合にはポジティブ・アクション・レポートが必要なことを示します。</p> <p>MQRO_NAN*: 処理が失敗した場合にはネガティブ・アクション・レポートが必要なことを示します。</p> <p>MQRO_COPY_MSG_ID_TO_CORREL_ID*: 生成されたレポートの相関 ID が、最初に発行された要求のメッセージ ID と等しくなければならないことを示します。</p>
ReplyToQueue	応答キューの名前。	応答メッセージが必要な場合は、このフィールドにコネクタ・プロパティー ReplyToQueue の値を指定します。
Persistence	メッセージの永続性。	MQPER_PERSISTENT*
Expiry	メッセージの存続時間。	MQEI_UNLIMITED*
* IBM で定義されている定数を示します。		

表 1 に示されているメッセージ・ヘッダーは、メッセージ本体の前に入ります。メッセージ本体は、データ・ハンドラーを使用してシリアルライズされたビジネス・オブジェクトです。

Report フィールドは、WebSphere Commerce からポジティブとネガティブの両方のレポートが要求されていることを示すように設定します。メッセージを発行したスレッドは、WebSphere Commerce が要求を処理できたかどうかを示す応答メッセージを待機します。

WebSphere Commerce は、アダプターから同期要求を受け取ると、ビジネス・オブジェクトのデータを処理し、レポート・メッセージを発行します (表 2、表 3、および表 4 を参照)。

表 2. 応答メッセージの記述子ヘッダー (MQMD)

フィールド	説明	値
Format	フォーマットの名前。	変換プロパティーで定義されている busObj の入力フォーマット。
MessageType	メッセージのタイプ。	MQMT_REPORT*
* IBM で定義されている定数を示します。		

表 3. 応答メッセージの生成

動詞	フィードバック・フィールド	メッセージ本体
Create、Update、Delete	SUCCESS VALCHANGE	(オプション) 変更を反映しているシリアルライズされたビジネス・オブジェクト。
	VALDUPES FAIL	(オプション) エラー・メッセージ。

表 4. WebSphere MQ のフィードバック・コードと ICS の応答値。

WebSphere MQ のフィードバック・コード	同等の ICS 応答 *
MQFB_PAN または MQFB_APPL_FIRST	SUCCESS
MQFB_NAN または MQFB_APPL_FIRST + 1	FAIL
MQFB_APPL_FIRST + 2	VALCHANGE
MQFB_APPL_FIRST + 3	VALDUPES
MQFB_APPL_FIRST + 4	MULTIPLE_HITS
MQFB_APPL_FIRST + 5	FAIL_RETRIEVE_BY_CONTENT
MQFB_APPL_FIRST + 6	BO_DOES_NOT_EXIST
MQFB_APPL_FIRST + 7	UNABLE_TO_LOGIN
MQFB_APPL_FIRST + 8	APP_RESPONSE_TIMEOUT (その結果、即時にコネクタ ーが終了します)
MQFB_NONE	応答メッセージにフィードバック・コードが指定されてい ない場合にコネクタが受け取る情報
* 詳細については、「コネクタ開発ガイド」を参照してください。	

ビジネス・オブジェクトを処理できる場合、アプリケーションは、フィードバック・フィールドを MQFB_PAN (または特定の ICS 値) に設定したレポート・メッセージを作成します。必要に応じて、アプリケーションは、メッセージ本体に、変更が加えられているシリアルライズされたビジネス・オブジェクトを取り込みます。ビジネス・オブジェクトを処理できない場合、アプリケーションは、フィードバック・フィールドを MQFB_NAN (または特定の ICS 値) に設定したレポート・メッセージを作成し、必要に応じてメッセージ本体にエラー・メッセージを組み込みます。いずれの場合も、アプリケーションは、メッセージの correlationID フィールドを、アダプター・メッセージの messageID に設定し、replyTo フィールドで指定されたキューに、そのメッセージを発行します。

応答メッセージを取得すると、アダプターは、応答の correlationID を要求メッセージの messageID と突き合わせます。次に、要求の発行元スレッドに通知します。応答のフィードバック・フィールドに応じて、アダプターは、メッセージ本体にビジネス・オブジェクトとエラー・メッセージのいずれが入っているかを判断します。予期されたビジネス・オブジェクトがメッセージ本体にない場合、アダプターは単に、Request 操作で InterChange Server が最初に発行したものと同一ビジネス・オブジェクトを戻します。予期されたエラー・メッセージがメッセージ本体にない場合、応答コードと共に汎用のエラー・メッセージが InterChange Server に戻されます。

カスタム・フィードバック・コードの作成: WebSphere MQ フィードバック・コードを拡張して、表 4 に示すデフォルトの解釈をオーバーライドするには、コネクタ・プロパティー FeedbackCodeMappingMO を指定します。このプロパティーによ

り、ICS 固有の戻り状況値がすべて WebSphere MQ のフィードバック・コードにマッピングされるメタオブジェクトを作成することができます。

フィードバック・コードに割り当てられた戻り状況は、InterChange Server に渡されません。詳細については、33 ページの『FeedbackCodeMappingMO』を参照してください。

Retrieve、Exists、Retrieve by Content

Retrieve、Exists、Retrieve by Content の動詞を使用するビジネス・オブジェクトは、同期転送しかサポートしていません。コネクタがこれらの動詞を使用するビジネス・オブジェクトを処理する方法は、Create、Update、Delete で定義されている同期転送の場合と同じです。ただし、Retrieve、Exists、Retrieve by Content の動詞を使用している場合は、responseTimeout と replyToQueue が必要です。さらに、Retrieve 動詞と Retrieve by Content 動詞の場合は、トランザクションを完了するために、メッセージ本体にシリアルライズされたビジネス・オブジェクトを取り込む必要があります。

表 5 に、これらの動詞の応答メッセージを示します。

表 5. 応答メッセージの生成

動詞	フィードバック・フィールド	メッセージ本体
Retrieve または RetrieveByContent	FAIL	(オプション) エラー・メッセージ。
	FAIL_RETRIEVE_BY_CONTENT	
	MULTIPLE_HITS SUCCESS	シリアルライズされたビジネス・オブジェクト。
Exist	FAIL	(オプション) エラー・メッセージ。
	SUCCESS	

共通の構成タスク

インストールしたアダプターは、始動する前に構成する必要があります。このセクションでは、一般に開発者が実行する必要がある構成作業と始動作業の概要を紹介します。

アダプターのインストール

インストールしなければならない要素と場所については、17 ページの『第 2 章 コネクタのインストールおよび構成』を参照してください。

コネクタ・プロパティの構成

コネクタには、標準構成プロパティとコネクタ固有の構成プロパティの 2 種類の構成プロパティがあります。これらのプロパティの中には、デフォルト値のまま、変更する必要がないものもあります。また、コネクタを実行する前にこれらのプロパティの値を設定しなければならないものもあります。詳細については、17 ページの『第 2 章 コネクタのインストールおよび構成』を参照してください。

Adapter for WebSphere Commerce のコネクタ・プロパティを構成する場合は、以下のことを確認してください。

- コネクタ・プロパティ `HostName` に指定した値が、WebSphere MQ サーバーのホストの値と一致している。
- コネクタ・プロパティ `Port` に指定した値が、アダプターで使用するキュー・マネージャーのリスナーのポートの値と一致している。
- コネクタ・プロパティ `Channel` に指定した値が、アダプターで使用するキュー・マネージャーのサーバー接続チャンネルと一致している。
- キューに指定した値が、キューを作成するときに使用した名前と一致している。
- コネクタ・プロパティ `InputQueue`、`InProgressQueue`、`ArchiveQueue`、`ErrorQueue`、`UnsubscribeQueue` のキュー URI が有効で、実際に存在する。表 7 を参照してください。

通知不要の要求の送信

通知不要の要求を送信するようにアダプターを構成するには、次のようにします (これはデフォルトの非同期モードで、「fire and forget 型 (送信後削除型)」とも言います)。

- 送信する要求を表し、XML データ・ハンドラーと互換性がある、ビジネス・オブジェクトを作成します。
- 静的または動的いずれかのメタオブジェクトを使用して、ターゲット・キューとフォーマットを指定します。静的メタオブジェクトと動的メタオブジェクトについては、それぞれ 42 ページの『静的メタオブジェクト』と 47 ページの『動的子メタオブジェクト』を参照してください。
- メタオブジェクト (静的または動的) のプロパティ `ResponseTimeout` を `-1` に設定します。この場合、コネクタはビジネス・オブジェクトを発行しますが、戻り値を確認することはありません。

要求の送信と通知の取得

注: この方法を行うには、ICS からビジネス・オブジェクトを受け取ったときに WebSphere Commerce で実行するコマンドをカスタマイズする必要があります。このコマンドは、メッセージから「ReplyTo」キューを取得し、`ResponseTimeout` 間隔内に応答をそのキューに入れる、というものです。WebSphere Commerce でのコマンドの作成とカスタマイズについては、「WebSphere Commerce プログラマーズ・ガイド 5.4」を参照してください。

要求を送信して通知を取得するようにアダプターを構成するには、正の `ResponseTimeout` 値を指定して、アダプターが応答を待機する時間を示します。

この方法では、コネクタ・プロパティで `ReplyTo` キューを定義することも必要です。コネクタが応答メッセージに期待する情報の詳細については、9 ページの『同期の引き渡し』を参照してください。応答メッセージが、ここに挙げられている要件を満たさないと、コネクタは、エラーを報告したり、異常終了して応答メッセージを認識できなかったりすることがあります。42 ページの『メタオブジェクト属性の構成』のセクション、および 55 ページの『第 3 章 ビジネス・オブジェクトの処理』も参照してください。

静的メタオブジェクトの構成

静的メタオブジェクトには、ビジネス・オブジェクトと、コネクターがビジネス・オブジェクトを処理する方法について指定した、アプリケーション固有の情報が収められています。コネクターは、始動時に、静的メタオブジェクトからビジネス・オブジェクトを処理するために必要な情報をすべて取得します。

インプリメンテーション時に、各種ビジネス・オブジェクトの送信先のキューが分かっている場合は、静的メタオブジェクトを使用します。このオブジェクトを作成、構成するには、次のようにします。

- 42 ページの『静的メタオブジェクト』のステップに従います。
- コネクター固有のプロパティ `DataHandlerConfigMO` で静的メタオブジェクトの名前を指定して、コネクターが静的メタオブジェクトをサブスクライブしていることを確認します。詳細については、31 ページの『コネクター固有のプロパティ』を参照してください。

動的メタオブジェクトの構成

コネクターが、シナリオに応じてビジネス・オブジェクトの処理方法を変更する必要がある場合は、動的メタオブジェクトを使用します。これは、ビジネス・オブジェクトに追加する子オブジェクトです。動的メタオブジェクトにより、コネクターは (ランタイムで) 要求を処理する方法を認識します。コネクターにビジネス・オブジェクトを処理するために必要な情報をすべて提供する静的メタオブジェクトと異なり、動的メタオブジェクトが提供するの、特定のシナリオで要求を処理する場合に必要なロジックの追加要素だけです。動的メタオブジェクトを作成、構成するには、次のようにします。

- 動的メタオブジェクトを作成し、子として要求ビジネス・オブジェクトに追加します。
- 動的メタオブジェクトにターゲット・キューやメッセージ・フォーマットなどの情報を取り込む追加ロジックを使用してコラボレーションをプログラミングし、コネクターに発行します。

コネクターは、動的メタオブジェクトをチェックし、そのメタオブジェクトの情報を使用して、ビジネス・オブジェクトの処理方法を判断します。詳細については、47 ページの『動的子メタオブジェクト』を参照してください。

MQMD フォーマットの構成

MQMD は、メッセージ記述子です。MQMD には、アプリケーション間でメッセージをやり取りする場合に、アプリケーション・データと一緒に渡される制御情報が収められています。静的または動的いずれかのメタオブジェクトで、MQMD 属性 `OutputFormat` の値を指定する必要があります。詳細については、9 ページの『Create、Update、Delete』を参照してください。

キュー URI の構成

コネクターで使用するキューを構成するには、次のようにします。

- すべてのキューを URI (Uniform Resource Identifier) として指定します。構文は、以下のとおりです。

```
queue://<InterChangeServerName.queue.manager>/<actual queue>
```

- コネクタ固有の構成プロパティで、キュー・マネージャーのホストを指定します (表 7 を参照)。
- ターゲット・アプリケーションが MQMD ヘッダーのみを処理し、JMS クライアントが使用する拡張 MQRFH2 ヘッダーを処理できない場合は、キュー URI に ?targetClient=1 を付加します。詳細については、40 ページの『キューの URI (Uniform Resource Identifiers)』、および WebSphere MQ のプログラミング・ガイドを参照してください。

XML データ・ハンドラーの構成

WebSphere Commerce でアダプターを使用するには、XML データ・ハンドラーが必要です。データ・ハンドラーを構成する方法は 2 つあります。

- コネクタ固有のプロパティ `DataHandlerClassName` で、データ・ハンドラーのクラス名を指定します。詳細については、31 ページの『コネクタ固有のプロパティ』を参照してください。
- コネクタ固有のプロパティ `DataHandlerMimeType` で MIME タイプを指定し、`DataHandlerConfigMO` でその MIME タイプの構成を定義するデータ・ハンドラーのメタオブジェクトを指定します。詳細については、表 7 および「*IBM WebSphere InterChange Server* データ・ハンドラー・ガイド」を参照してください。

始動スクリプトの変更

コネクタの始動方法については、17 ページの『第 2 章 コネクタのインストールおよび構成』を参照してください。始動する前に、コネクタ・プロパティを構成する必要があります。また、始動ファイルも変更する必要があります。

- クライアント・ライブラリーのロケーションが指定されるように、`start_connector` スクリプトを変更してください。複数のバージョン、または現在使用している WebSphere MQ サーバーに対応しないバージョンのクライアント・ライブラリーをインストールしないようにしてください。詳細については、52 ページの『始動ファイルの構成』を参照してください。

第 2 章 コネクタのインストールおよび構成

この章では、アダプターをインストールして構成する方法、および WebSphere Commerce アプリケーションでコネクタを使用できるようにする方法について説明します。この章には、以下のトピックが記載されています。

- 17 ページの『アダプター環境』
- 18 ページの『前提条件となる作業』
- 30 ページの『アダプターおよび関連ファイルのインストール』
- 30 ページの『アダプターの構成』
- 35 ページの『保証付きイベント・デリバリー機能の使用可能化』
- 40 ページの『キューの URI (Uniform Resource Identifiers)』
- 42 ページの『メタオブジェクト属性の構成』
- 17 ページの『ブローカーの互換性』
- 52 ページの『始動ファイルの構成』
- 53 ページの『コネクタの開始』
- 54 ページの『コネクタの停止』

アダプター環境

アダプターをインストール、構成、使用する前に、環境要件を理解しておく必要があります。環境要件は、以下のセクションでリストされています。

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

ブローカーの互換性

アダプターが使用するアダプター・フレームワークは、アダプターと通信する統合ブローカーのバージョンとの互換性を備えている必要があります。Adapter for WebSphere Commerce バージョン 2.5 は、以下のアダプター・フレームワークと統合ブローカーでサポートされています。

- **アダプター・フレームワーク:**
WebSphere Business Integration Adapter Framework バージョン 2.3.1、および 2.4
- **統合ブローカー:**
 - WebSphere InterChange Server、バージョン 4.1.1、4.2、4.2.1、4.2.2
 - 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
(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: 統合ブローカーおよびその前提条件のインストールに関する説明については、以下のガイドを参照してください。

WebSphere InterChange Server (ICS) については、「*IBM WebSphere InterChange Server システム・インストール・ガイド (UNIX 版)*」または「*IBM WebSphere InterChange Server システム・インストール・ガイド (Windows 版)*」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。

アダプターのプラットフォーム

このアダプターは、次のソフトウェアでサポートされています。

オペレーティング・システム:

以下のアプリケーション・プラットフォームのいずれか:

- AIX 5.1、AIX 5.2
- Solaris 7.0、Solaris 8.0
- HP UX 11.0、HP UX 11i
- Windows 2000
- iSeries
- Linux

データベース:

- DB2

サード・パーティー・ソフトウェア:

- WebSphere Commerce バージョン 5.4 および 5.5

アダプターの依存関係

このアダプターには以下の依存関係があります。

- WebSphere Application Server ライブラリー/API が必要。

グローバル化

このアダプターは DBCS (2 バイト文字セット) に対応していますが、翻訳されていません。

前提条件となる作業

このセクションでは、アダプターをインストールして実行する前に、WebSphere Commerce およびその他のソフトウェアに対して実行しなければならないインストール作業と構成作業について説明します。

実行しなければならない作業は、以下のとおりです。

1. WebSphere Commerce のインストールと構成
2. Commerce 拡張パックのインストールと構成
3. WebSphere Commerce ストアのパブリッシュ
4. WebSphere MQ キューの作成と構成
5. WebSphere Application Server の JMS 設定の構成
6. WebSphere Commerce 内の JMS ConnectionSpec の構成
7. WebSphere Commerce の JVM 設定の更新
8. WebSphere Commerce Adapter の使用可能化

WebSphere Commerce のインストールと構成

WebSphere Commerce バージョン 5.4 Business Edition (修正パッケージ 2) をインストールします。インストール手順とインストール後の構成については、製品に付属のマニュアルを参照してください。WebSphere Commerce メッセージング・システムは、メッセージを処理してバックエンド・システムとやり取りするために装備されています。

CMDREG テーブルを更新する必要があります。これは、XML メッセージ・フォーマットを使用するための、WebSphere Commerce データベース内のコマンド・レジストリー・テーブルです。

Commerce 拡張パックのインストール

Commerce 拡張パックをインストールするには、URL <http://www.ibm.com/software/commerce/epacks> から Commerce 拡張パック・ドライバーをダウンロードし、readme.txt ファイルの指示に従います。

ストアのパブリッシュ

このアダプターは、WebSphere Commerce の既存のパブリッシュ済みストアと併用することができます。また、新しいストアを作成することもできます。

WebSphere MQ キューの構成

アダプターの使用に必要な WebSphere MQ キューの構成は、WebSphere Commerce と IBM WebSphere InterChange Server をインストールする場合の接続形態により、ある程度異なります。サポートされている接続形態は、以下のとおりです。

- マシンが 1 台

WebSphere Commerce と IBM WebSphere InterChange Server とアダプターがすべて、同じマシンにインストールされています。

- マシンが 2 台で、キュー・マネージャーが 2 つ

WebSphere Commerce が 1 台のマシンにインストールされ、IBM WebSphere InterChange Server とコネクターがもう 1 台のマシンにインストールされています。マシンごとに別々のキュー・マネージャーが使用されています。

- マシンが 2 台で、キュー・マネージャーが 1 つ

WebSphere Commerce が 1 台のマシンにインストールされ、IBM WebSphere InterChange Server とコネクタがもう 1 台のマシンにインストールされています。同じキュー・マネージャーを使用して、両方のマシンのキューを管理しています。

マシンが 1 台の場合

この接続形態では、WebSphere Commerce と ICS と Adapter for WebSphere Commerce がすべて、1 台のマシンにインストールされています。単一のキュー・マネージャーが、ソリューションで使用するすべての WebSphere MQ キューを処理します。ICS をインストールしたときにセットアップしたキュー・マネージャーを使用することをお勧めします。

この接続形態では、以下の役割を行うキューが必要です。

- Inbound Queue (インバウンド・キュー)
これは、WebSphere Commerce が必要とするキューです。ただし、アダプターは、このソリューションではこのキューを利用しません。
- Parallel Inbound Queue (並列インバウンド・キュー)
これは、WebSphere Commerce が必要とするキューです。ただし、アダプターは、このソリューションではこのキューを利用しません。
- Serial Inbound Queue (逐次インバウンド・キュー)
ICS から WebSphere Commerce に送信されるメッセージの受信用。
- Outbound Queue (アウトバウンド・キュー)
WebSphere Commerce から ICS へのメッセージの送信用。
- InProgress Queue (処理中キュー)
WebSphere Commerce から ICS に送信される有効なメッセージのオリジナル・バージョンは、アダプターが処理を完了するまでここに保管されます。処理が完了すると、オリジナルのメッセージはローカルのアーカイブ・キューに移されます。
- Archive_Queue (アーカイブ・キュー)
メッセージがアダプターにより完全に処理されて、WebSphere Commerce から ICS に送信されると、メッセージのオリジナル・バージョンはここに保管されます。
- Unsubscribed_Queue (アンサブスクライブ・キュー)
メッセージが正常に解析されたが、アダプターでサポートされるビジネス・オブジェクトに対応していないと、ここに保管されます。
- ICS_Error_Queue (ICS エラー・キュー)
メッセージは、ビジネス・オブジェクトに正常に変換されずに ICS に送信されると、ここに保管されます。
- WCS_Error_Queue (WCS エラー・キュー)
WebSphere Commerce で正常に処理できないメッセージが保管されます。
- ReplyTo Queue (応答先キュー)
同期データ交換用にセットアップされた構成でのみ使用されます。

マシンが 1 台の接続形態では、上記のキューはすべてローカルです。キューを手動で作成する場合は、そのキューに割り当てる名前を選択します。このソリューションでバッチ・ファイルを使用する場合 (以下を参照) は、バッチ・ファイルが、事前に割り当て済みの名前を使ってキューを作成します。

Windows 環境でアダプターを使用している場合は、バッチ・ファイルを使用して、マシンが 1 台の接続形態に適したキューを生成することができます。このファイルは、製品パッケージと一緒に、IBM ICS のインストールで使ったルート・ディレクトリ内の ¥Connector¥WebSphereCommerce¥Utilities サブディレクトリにインストールされます。このバッチ・ファイルを使用してキューを作成するには、以下のように、ファイル `ConfigureWebSphereCommerceAdapter.bat` を実行します。

コマンド・プロンプトに、以下のように入力します。

```
ConfigureWebSphereCommerceAdapter  
<InterChangeServerName>.queue.manager
```

ここで、`<InterChangeServerName>` は WebSphere InterChange Server の名前です。

これで、`InterChange ServerName.queue.manager` という名前のキュー・マネージャーが作成され、必要な WebSphere MQ キューが作成されます。バッチ・ファイルによって作成される、作成済みキューの名前は、以下のとおりです。

WC_MQCONN.IN_PROGRESS: アダプター用の InProgress Queue。

WC_MQCONN.ERROR: アダプター用の ICS Error Queue。

WC_MQCONN.ARCHIVE: アダプター用の Archive Queue。

WC_MQCONN.REPLY: アダプター用の Reply-To_Queue。

WC_MQCONN.UNSUBSCRIBED: アダプター用の UnSubscribed Queue。

WCS_Serial_Inbound: WebSphere Commerce 用の Serial Inbound Queue。25 ページの『WebSphere アプリケーション・サーバーの JMS 設定の構成』に示すように、WebSphere Commerce に対して定義されている JMS キュー名と一致しなければなりません。

WCS_Outbound: WebSphere Commerce 用の Outbound Queue。25 ページの『WebSphere アプリケーション・サーバーの JMS 設定の構成』に示すように、WebSphere Commerce に対して定義されている JMS キュー名と一致しなければなりません。

WCS_Parallel_Inbound: WebSphere Commerce 用の Parallel Inbound Queue。25 ページの『WebSphere アプリケーション・サーバーの JMS 設定の構成』に示すように、WebSphere Commerce に対して定義されている JMS キュー名と一致しなければなりません。

WCS_Error: WebSphere Commerce 用の Error Queue。25 ページの『WebSphere アプリケーション・サーバーの JMS 設定の構成』に示すように、WebSphere Commerce に対して定義されている JMS キュー名と一致しなければなりません。

WCS_Inbound: WebSphere Commerce 用の Inbound Queue。25 ページの『WebSphere アプリケーション・サーバーの JMS 設定の構成』に示すように、WebSphere Commerce に対して定義されている JMS キュー名と一致しなければなりません。

マシンが 2 台で キュー・マネージャーが 2 つの場合

この接続形態では、WebSphere Commerce が 1 台のマシンにインストールされ、IBM WebSphere ICS と Adapter for WebSphere Commerce がもう 1 台のマシンにインストールされています。

WebSphere MQ は、マシンごとにインストールしなければならないので、インストールごとに別々のキュー・マネージャーを使用します。以下に、各マシンで作成するキューを示します。

注: この表では、キューの名前はそれぞれのキューの役割を表していますが、WebSphere Commerce システムで使用している JMS キューの名前と一致していれば、どのようなキュー名を設定しても構いません。

この表で、接頭部 WCS は、WebSphere Commerce システムがインストールされているマシンで作成され、そのマシンに常駐するキュー・マネージャーで管理されるキューを示します。

接頭部 ICS は、ICS とコネクタがインストールされているマシンで作成され、そのマシンに常駐するキュー・マネージャーで管理されるキューを示します。

WebSphere Commerce マシン上のキュー	ICS マシン上のキュー
WCS_Outbound Queue	ICS_Inbound queue
WebSphere Commerce から ICS へのメッセージの送信用。このキューは、リモート・キューとして ICS マシン上の ICS_Inbound を指す、リモート・キュー定義として作成されます。	WebSphere Commerce から ICS に送信されるメッセージの受信用。
WCS_Serial Inbound queue	ICS_Outbound queue
ICS から WebSphere Commerce に送信されるメッセージの受信用。	ICS から WebSphere Commerce へのメッセージの送信用。このキューは、リモート・キューとして WebSphere Commerce マシン上の WCS_Serial Inbound を指す、リモート・キュー定義として作成されます。
To ICS	To WebSphere Commerce
WebSphere Commerce システムから ICS への送信キュー。	ICS から WebSphere Commerce システムへの送信キュー。
WCS_Error_queue	ICS_Error_queue
WebSphere Commerce で正常に処理できないメッセージが保管されます。	メッセージは、ビジネス・オブジェクトに正常に変換されないと、ここに保管されます。アダプターは、並列インバウンド・キューを利用しません。
WCS_Parallel Inbound	

ICS_InProgress queue

WebSphere Commerce から ICS に送信される有効なメッセージのオリジナル・バージョンは、アダプターが処理を完了するまでここに保管されます。処理が完了すると、オリジナルのメッセージはローカルのアーカイブ・キューに移されます。

ICS_Archive_queue

メッセージがアダプターにより完全に処理されて、WebSphere Commerce から ICS に送信されると、メッセージのオリジナル・バージョンはここに保管されます。

ICS_Unsubscribed_queue

メッセージがビジネス・オブジェクトに正常に変換されたが、アダプターでサポートされるビジネス・オブジェクトに対応していないと、ここに保管されます。

2 つのシステム間の通信を確立するには、チャンネルと送信キューを使用します。

この接続形態では、各マシンごとに以下の役割を実行するチャンネルを作成しなければなりません。

WebSphere Commerce

マシン上のチャンネル

Sender_WCS
Receiver_ICCS

ICS マシン上のチャンネル

Sender_ICCS
Receiver_WCS

チャンネルの作成: 以下の手順では、別々のマシンとキューに対応付けるように、特定のサーバーとキュー・マネージャーの名前を指定しています。チャンネルは、正しいキューが参照し合っていることを確認するためのものです。キューの「ローカル」バージョンは、実際の情報を保持するためのものです。

WebSphere Commerce マシンで、以下の構成作業を行います。

注: ここで使用するチャンネルの名前は、単なる例です。

1. WebSphere MQ Explorer を使用して、WebSphere Commerce システムで 2 つのチャンネルを作成します。1 つは、'WCS' という送信側チャンネル、もう 1 つは 'ICS' という受信側チャンネルです。
2. ローカル・キューを作成します (例えば、'ToICSSystem' という名前を使用します)。
3. ToICSSystem キューを送信キューとして設定します。
4. WCS_Outbound Queue について、以下のプロパティを設定します。
 - a. リモート・キュー名 ICS_Inbound リモート・キュー・マネージャー名 ICS_server_name.queue.manager。例えば、ICS.queue.manager とします。

- b. ステップ 2 で作成した ‘ToICSSystem’ について、送信キュー名のプロパティを設定します。
5. 送信側チャンネルを構成するには、以下のようにします。
 - a. IP アドレスとポートからなる接続名を指定します (例えば、9.182.12.235(1414))。ここで、9.182.12.235 は、ICS が実行されているマシンの IP アドレスで、1414 は、デフォルトのリスナー・ポートです。
 - b. 送信キューの名前を ‘ToICSSystem’ として指定します。

これで、WebSphere Commerce マシンの構成作業は終わりです。

次に、ICS マシンで以下の構成作業を行います。

1. WebSphere MQ Explorer を使用して、‘ICS’ という送信側チャンネルと ‘WCS’ という受信側チャンネルの 2 つのチャンネルを作成します。

注: WebSphere Business Integration システムの送信側チャンネルの名前は、WebSphere Commerce の受信側チャンネルの名前と一致していなければなりません。WebSphere Business Integration システムの受信側チャンネルの名前は、WebSphere Commerce の送信側チャンネルの名前と一致していなければなりません。

2. 新しいローカル・キュー (例えば、‘ToWCSSystem’) を作成します。
ToWCSSystem キューを送信キューとして設定します。
3. WebSphere Business Integration システムにリモート定義キューを作成します。このリモート定義キューは、コネクタ・コンポーネントで出力キューとして使用してください。以下のプロパティを設定します。
 - a. リモート・キュー名 WCS_SerialInbound
 - b. リモート・キュー・マネージャー名 <wcssystem_Q_manager_name>。例えば、QM_wcsfvt3 とします。
 - c. 送信キュー名のプロパティを ‘ToWCSSystem’ に設定します。
4. 送信側チャンネルを構成するには、以下のようにします。
 - a. IP アドレスとポートからなる接続名を指定します (例えば、9.182.12.18(1414))。ここで、9.182.12.18 は、WebSphere Commerce が実行されているマシンの IP アドレスで、1414 はデフォルトのリスナー・ポートです。
 - b. 送信キューの名前を ‘TOWCSSystem’ として指定します。

WebSphere Commerce マシンと ICS マシンの両方で WebSphere MQ キューとチャンネルの構成が終了したら、受信側チャンネル、送信側チャンネルの順に始動します。

マシンが 2 台でキュー・マネージャーが 1 つの場合

この接続形態では、WebSphere Commerce が 1 台のマシンにインストールされ、ICS とアダプターがもう 1 台のマシンにインストールされています。WebSphere MQ のインスタンスは 1 つしか実行されていないので、両方のマシンで使用するキューは単一のキュー・マネージャーで管理されます。このシナリオでは、ローカル・キューしか使用しません。

この接続形態では、以下の役割を行うキューが必要です。

- Inbound Queue (インバウンド・キュー)

これは、WebSphere Commerce が必要とするキューです。ただし、アダプターは、このソリューションではこのキューを利用しません。

- **Serial Inbound Queue (逐次インバウンド・キュー)**
ICS から WebSphere Commerce へのメッセージの送信用。
- **Outbound Queue (アウトバウンド・キュー)**
WebSphere Commerce から ICS に送信されるメッセージの受信用。
- **InProgress Queue (処理中キュー)**
WebSphere Commerce から ICS に送信される有効なメッセージのオリジナル・バージョンは、アダプターが処理を完了するまでここに保管されます。処理が完了すると、オリジナルのメッセージはローカルのアーカイブ・キューに移されます。
- **Archive_Queue (アーカイブ・キュー)**
メッセージがアダプターにより完全に処理されて、WebSphere Commerce から ICS に送信されると、メッセージのオリジナル・バージョンはここに保管されます。
- **Unsubscribed_Queue (アンサブスクライブ・キュー)**
メッセージが正常に解析されたが、アダプターでサポートされるビジネス・オブジェクトに対応していないと、ここに保管されます。
- **WebSphere Commerce Error Queue (WebSphere Commerce エラー・キュー)**
- **ICS_Error_Queue (ICS エラー・キュー)**
メッセージは、ビジネス・オブジェクトに正常に変換されずに ICS に送信されると、ここに保管されます。
- **WCS_Error_Queue (WCS エラー・キュー)**
WebSphere Commerce で正常に処理できないメッセージが保管されます。
- **ReplyTo Queue (応答先キュー)**
同期データ交換用にセットアップされた構成でのみ使用されます。

WebSphere アプリケーション・サーバーの JMS 設定の構成

Java Messaging Service Connection Factory と JMS キューを作成して WebSphere MQ で使用できるように、WebSphere Application Server (WAS) を構成する必要があります。以下の手順を実行します。

1. コマンド・プロンプトから、以下のことを実行します。

- a. 1 行に以下のコマンドを入力し、classpath 変数を更新します。

```
set classpath= %classpath%;  
MQ_install_path¥java¥lib¥com.ibm.mqjms.jar;  
MQ_install_path¥java¥lib¥com.ibm.mq.jar;  
MQ_install_path¥java¥lib¥com.ibm.mq.iopp.jar;  
MQ_install_path¥java¥lib¥com.ibm.ibmorb.jar;WAS_install_path¥lib¥ns.jar
```

MQ_install_path WebSphere MQ のインストール・パス。

WAS_install_path WebSphere Application Server のインストール・パス。

- b. 以下のコマンドを入力して、新しい環境変数 `MQ_JAVA_INSTALL_PATH` を追加します。

```
set MQ_JAVA_INSTALL_PATH=MQ_install_path¥java
```

MQ_install_path WebSphere MQ のインストール・パス。

- c. 以下のコマンドを入力して、WebSphere Application Server に付属の JDK (Java Development Kit) を使用するように環境を更新します。

```
set PATH = WAS_Intall_Path¥Java¥bin;%PATH%
```

WAS_install_path WebSphere Application Server のインストール・パス。

2. WebSphere Application Server が実行されていて、上記のステップ 1 で定義した正しいクラスパス変数と環境変数が追加されていることを確認します。また、`java -version` を実行して、バージョンを `WAS_Install_Path¥Java¥bin` に示されている情報と突き合わせて、使用している JDK が WAS の JDK であることも確認します。

3. `MQ_install_path¥java¥bin` ディレクトリーで、`JMSAdmin.config` ファイルを開き、以下の値を設定します。

```
INITIAL_CONTEXT_FACTORY=com.ibm.ejs.ns.jndi.CNInitialContextFactory  
PROVIDER_URL=iiop://localhost:900
```

上記の値は、WebSphere Commerce と WebSphere MQ が同じマシンにインストールされていることを前提とします。

```
SECURITY_AUTHENTICATION=none
```

4. コマンド行入力として `JMSAdmin.config` ファイルを指定し、`JMSAdmin` プログラムを実行します。

```
コマンド・プロンプト:> JMSAdmin -cfg JMSAdmin.config -t -v
```

このコマンドを実行すると、WebSphere Application Server が提供する JNDI (Java Naming and Directory Interface) サービスを検索できるようになります。

`InitCtx>` プロンプトが表示されるので、これを使用して `JMS` 管理コマンドを実行することができます。

5. `QueueConnectionFactory` を登録し、以下のコマンドを入力してコード化文字セット ID を設定します。

- `define qcf (JMS_QueueConnectionFactory) qmanager (Your_QueueManager_Name)`
- `alter qcf (JMS_QueueConnectionFactory) ccsid(1208)`

ここで、`JMS_QueueConnectionFactory` は、`MQQueueConnectionFactory` `JMS` オブジェクトの名前です。

上記の一連のコマンドを実行すると、このキュー・コネクション・ファクトリーの項目が WebSphere Application Server データベースの `BINDINGBEANTBL` テーブル下に作成されます。これらのオブジェクトは、WebSphere Application Server データベースに登録されます。

6. WebSphere MQ キューに対して設定した名前と使用している WebSphere MQ キュー・マネージャーにマップするように、以下の `JMSQueue` を定義します。`JMSQueue` の名前は、ユーザー要件に合わせてカスタマイズできますが、定義する `JMSQueue` に対応する WebSphere MQ の名前は、大文字小文字を含め、WebSphere MQ で設定したキュー名と正確に一致しなければなりません。このアダプターに付属のバッチ・ファイルを使用して、単一マシンの接続形態に適した WebSphere MQ キューを作成している場合は、以下の表に示すように、バッチ・

ファイルで生成されたこれら保証済みの WebSphere MQ キュー名を、定義する JMS キューに対応する値として使用してください。

JMS キューを定義する構文は、以下のとおりです。

注: マシンが 2 台でキュー・マネージャーが 2 つの場合と同じように、アウトバウンド・キューにリモート・キュー定義を使用している場合は、JMS_Outbound_Queue をローカルの WebSphere MQ キューに対して定義しないでください。リモート・キュー定義を使用している場合、アウトバウンド・キューの構文は、以下のようになります。define
q(JMS_Outbound_Queue)qmanager(Your_Queue_Manager_Name)

```
define q(JMS_Outbound_Queue)qmanager
  (Your_Queue_Manager_Name)
  queue(Your_Outbound_QueueName)

define q(JMS_Inbound_Queue)qmanager
  (Your_Queue_Manager_Name)
  queue(Your_Inbound_QueueName)

define q(JMS_Parallel_Inbound_Queue)qmanager
  (Your_Queue_Manager_Name)queue
  (Your_Parallel_Inbound_Queue_Name)

define q(JMS_Serial_Inbound_Queue)qmanager
  (Your_Queue_Manager_Name)queue
  (Your_Serial_Inbound_Queue_Name)

define q(JMS_Error_Queue)qmanager
  (Your_Queue_Manager_Name)
  queue (Your_Error_Queue_Name)
```

表 6. JMS キュー名の定義

<i>Your_Outbound_QueueName</i>	アウトバウンド・キューに対して作成する WebSphere MQ キュー。デフォルトでは、これは、アダプターが WebSphere Commerce から ICS に渡すメッセージを選ぶときにポーリングするキューです。バッチ・ファイルで作成されるデフォルトの WebSphere MQ キュー設定では、この値は WCS_Outbound になります。
<i>Your_Serial_Inbound_Queue</i>	逐次インバウンド・キューに対して作成する MQ キュー。これは、WebSphere Commerce の MQ アダプターが、ICS から Websphere Commerce に送信するメッセージを入れるキューです。バッチ・ファイルで作成されるデフォルトの WebSphere MQ キュー設定では、この値は WCS_Serial_Inbound になります。
<i>Your_Parallel_Inbound_Queue_Name</i>	これは、並列インバウンド・キューに対して作成する WebSphere MQ キューです。
<i>Your_Error_Queue_Name</i>	エラー・キューに対して作成する WebSphere MQ キュー。メッセージでエラーが発生すると、WebSphere Commerce MQ Adapter はここにメッセージを送信します。バッチ・ファイルで作成されるデフォルトの WebSphere MQ キュー設定では、この値は WCS_Error になります。

表 6. JMS キュー名の定義 (続き)

<p><i>Your_Queue_Manager_Name</i></p>	<p>WebSphere Commerce システムのユーザー設定で、WebSphere MQ キューを処理するキュー・マネージャーの名前。バッチ・ファイル <code>ConfigureAdapterQueues.bat</code> (19 ページの『WebSphere Commerce のインストールと構成』を参照) で作成されるセットアップなど、マシンが 1 台の一般的なセットアップでは、ICS に対して設定したキュー・マネージャーを使用して、WebSphere Commerce システムのキューも管理します。このようなセットアップでは、デフォルトは <code><InterchangeServerName>.queue.manager</code> になります。</p>
---------------------------------------	--

キューを作成し終わったら、JMSAdmin コンソールを使用して、アウトバウンド・キューとエラー・キューの以下のプロパティーを設定します。ここでは、JMS が固有の WebSphere MQ アプリケーションを扱うことを指定します。

- `alter q(JMSOutboundQueue) targclient(MQ)`
- `alter q(JMSErrorQueue) targclient(MQ)`

`end` と入力し、JMSAdmin ツールを終了します。これで、WebSphere Application Server で WebSphere Commerce が実行されるように Java Messaging Service を構成する作業は終わりです。

WebSphere Commerce 内での JMS ConnectionSpec の構成

注: JMSQueue 名と JMS Connection Factory は、インスタンス XML ファイル内で、Commerce Configuration Manager の `connectionSpec` セクションに入力した値と同じでなければなりません。詳細については、WebSphere Commerce Configuration Manager の「トランスポート (Transports)」セクションを参照してください。また、以下の説明も参照してください。

WebSphere Commerce 管理コンソールを始動します。サイト管理者としてログインし、「構成」セクションに移動して、「トランスポート (Transport)」オプションを選択します。トランスポートとして WebSphere MQ を選択し、状況をアクティブに変更します。管理コンソールからログアウトします。

WebSphere Commerce ソリューションでは、「WebSphere Commerce インストール・ガイド」に記載されているように、「ストア」の作成と使用が必要です。同資料の『サンプル・ストアのパブリッシュ (Publishing a Sample Store)』のセクションに従ってストアのパブリッシュを完了したら、今度はストア管理者として管理コンソールにログオンし、使用しているストアを選択します。「構成」セクションで、ストアに MQ Transport を追加します。これに対する項目は、STORETRANS テーブルに作成されます。

メッセージング・システムのトランスポート・アダプターを使用可能にするには、WebSphere Commerce Configuration Manager を立ち上げて、以下のことを実行します。

1. 「ホスト名」->「インスタンス」を選択して、「コンポーネント」フォルダーを開きます。
2. 「TransportAdapter」を選択します。
3. 「コンポーネントを使用可能にする (Enable Component)」チェック・ボックスを選択し、「適用」をクリックします。

以下のように、このインスタンスで connectionSpec に使用している値を使って、JMSQueue の名前と JMS Connection Factory を構成します。

1. 「ホスト名」->「インスタンス」を選択します。
2. 「トランスポート (Transports)」を選択し、「アウトバウンド (Outbound)」->「JMS」と展開します。
3. 「ConnectionSpec」を選択します。
4. 「ConnectionFactory」に、WAS の JMS 設定を構成したときに作成した名前を入力します。上記で作成した Inbound Queue、Error Queue、および Outbound Queue の名前を入力します。
5. 「適用」をクリックします。
6. 「インバウンド (Inbound)」->「JMSInbound CCF Connector」-「逐次 (Serial)」と展開します。
7. 「ConnectionSpec」を選択します。
8. ConnectionFactory 名、SerialInbound、Error、および Output の各 JMS キュー名を入力します。
9. 「適用」をクリックします。
10. 「インバウンド (Inbound)」-「JMSInbound CCF Connector」-「並列 (Parallel)」と展開します。
11. 「ConnectionSpec」を選択します。
12. ConnectionFactory、ParallelInbound、Error、Output の各 JMS キュー名を入力します。
13. 「適用」をクリックします。

Configuration Manager を終了します。

WebSphere Commerce の JVM 設定の更新

インスタンスに合わせて WebSphere Application Server クラスパスを更新し、その他の jar ファイル項目を追加する必要があります。このためには、WebSphere Application Server Advanced Administrative Console を開き、以下のことを実行します。

1. WebSphere Commerce インスタンスが実行されているホストを選択します。
2. 「WebSphere 管理可能ドメイン (WebSphere Administrative Domain)」を選択します。
3. 「ノード (Nodes)」を選択します。
4. ホスト名を選択します。
5. 「アプリケーション・サーバー (Application Servers)」を選択します。
6. 「WebSphere Commerce Server instance_name」を選択します (ここで、instance_name は WebSphere Commerce インスタンスの名前です)。

7. インスタンスの JVM 設定に進みます。
8. 「新規システム・プロパティの追加 (Add a new system property)」を選択します。
9. 以下のシステム・プロパティを入力します。

```
name= ws.ext.dirs value=MQ_INSTALL_PATH/java/lib
```

ここで、MQ_INSTALL_PATH は WebSphere WebSphere MQ のインストール・パスです。

10. WebSphere Application Server サービスを再始動し、すべての変更を有効にします。

アダプターに対する WebSphere MQ の使用可能化

WebSphere Commerce で、管理コンソール下の構成オプションを使用して、WebSphere Commerce と IBM WebSphere ICS のインプリメンテーションで使用しているアウトバウンドとインバウンドのメッセージング用の WebSphere MQ キューと通信するように WebSphere Commerce を構成します。必要に応じて、「*WebSphere Commerce* オンライン・ヘルプ」ガイドを参照してください。

アダプターおよび関連ファイルのインストール

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

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

アダプターの構成

コネクターには、標準構成プロパティとコネクター固有の構成プロパティの 2 種類の構成プロパティがあります。アダプターを実行する前に、これらのプロパティの値を設定してください。ICS を統合ブローカーとして使用するコネクターのプロパティを構成する場合は、Connector Configurator を使用します。詳細については、79 ページの『付録 B. Connector Configurator』を参照してください。

コネクターは、始動時に構成値を取得します。ランタイム・セッション時に、1 つ以上のコネクター・プロパティの値を変更することができます。AgentTraceLevel などのコネクター構成プロパティに対する変更は、即時に有効になります。その他のコネクター・プロパティに対する変更は、変更後にコンポーネントまたはシステムのいずれかを再始動する必要があります。プロパティが動的 (即時に有効になる) と静的 (コネクター・コンポーネントまたはシステムのいずれかの再始動が必要になる) のいずれであるかを判別するには、Connector Configurator の「更新メソッド」の列を参照してください。

標準コネクター・プロパティ

標準構成プロパティは、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の詳細については、61 ページの『付録 A. コネクターの標準構成プロパティ』を参照してください。

このアダプターが統合ブローカーとしてサポートしているのは InterChange Server (ICS) だけなので、このコネクターに関連する構成プロパティーは ICS だけに適用されます。

コネクター固有のプロパティー

コネクター固有の構成プロパティーでは、ランタイムにコネクターが必要とする情報を指定します。また、エージェントのコードを変更したり再ビルドしたりすることなく、アダプター内の静的情報やロジックを変更することもできます。

以下の表に、アダプターのコネクター固有の構成プロパティーをリストします。プロパティーの説明については、以下の各セクションを参照してください。

注: これらのプロパティーには、デフォルトのキュー名値も含まれます。これらの値は、ユーザー環境で実際に使用しているキュー名と一致するように変更する必要があります。

表 7. 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	必須
ApplicationPassword	ログイン・パスワード		いいえ
ApplicationUserName	ログイン・ユーザー ID		いいえ
ArchiveQueue	正常に処理されたメッセージのコピーの送信先のキュー	queue://<queue_manager_name>/WC_MQCONN.ARCHIVE	いいえ
Channel	MQ サーバーのコネクター・チャンネル		はい
ConfigurationMetaObject	構成メタオブジェクトの名前		はい
DataHandlerClassName	データ・ハンドラーのクラス名	com.crossworlds.DataHandlers.text.xml	いいえ
DataHandlerConfigMO	データ・ハンドラーのメタオブジェクト	MO_DataHandler_Default	はい
DataHandlerMimeType	ファイルの MIME タイプ	text/xml	いいえ
ErrorQueue	処理されていないメッセージのキュー	queue://<queue_manager_name>/WC_MQCONN.ERROR	いいえ
FeedbackCodeMappingMO	フィードバック・コードのメタオブジェクト		いいえ
HostName	WebSphere MQ サーバー		はい
InDoubtEvents	FailOnStartup Ignore LogError	Reprocess	いいえ
InputQueue	ポーリング・キュー	queue://<queue_manager_name>/WC_MQCONN.IN	いいえ
InProgressQueue	処理中イベント・キュー	queue://<queue_manager_name>/WC_MQCONN.IN_PROGRESS	はい
PollQuantity	InputQueue プロパティーで指定されている各キューから取得するメッセージの数	1	いいえ
Port	WebSphere MQ リスナーに対して設定されているポート		はい

表7. コネクタ固有のプロパティ (続き)

名前	指定可能な値	デフォルト値	必須
ReplyToQueue	アダプターが要求を発行したときの応答メッセージの引き渡し先のキュー	queue://<queue_manager_name>/WC_MQCONN.REPLYTO	いいえ
UnsubscribedQueue	アンサブスクライブされたメッセージの送信先のキュー	queue://<queue_manager_name>/WC_MQCONN.UNSUBSCRIBE	いいえ
UseDefaults	true または false	false	

ApplicationPassword

WebSphere MQ にログインする場合に UserID と併用するパスワード。

デフォルト値は None です。

ApplicationPassword がブランクまたは取り除かれていると、アダプターは WebSphere MQ が提供するデフォルトのパスワードを使用します。

ApplicationUserName

WebSphere MQ にログインする場合に Password と併用するユーザー ID。

デフォルト値は None です。

ApplicationUserName がブランクまたは取り除かれていると、アダプターは WebSphere MQ が提供するデフォルトのユーザー ID を使用します。

ArchiveQueue

正常に処理されたメッセージのコピーの送信先のキュー。

デフォルト値は queue://<queue_manager_name>/WC_MQCONN.ARCHIVE です。

Channel

アダプターが WebSphere MQ と通信する際に使う MQ サーバー・アダプターのチャンネル。

デフォルト値は None です。

Channel がブランクまたは取り除かれていると、アダプターは WebSphere MQ が提供するデフォルトのサーバー・チャンネルを使用します。

ConfigurationMetaObject

コネクターの構成情報が収められている静的メタオブジェクトの名前。

デフォルト値は None です。

DataHandlerClassName

メッセージからビジネス・オブジェクトへの変換およびビジネス・オブジェクトからメッセージへの変換に使用する、データ・ハンドラー・クラス。

デフォルト値は com.crossworlds.DataHandlers.text.xml です。

DataHandlerConfigMO

構成情報を提供するためにデータ・ハンドラーに渡すメタオブジェクト。

デフォルト値は `MO_DataHandler_Default` です。

DataHandlerMimeType

特定の MIME タイプに基づいてデータ・ハンドラーを要求することができます。

WebSphere Commerce でアダプターを使用する場合は XML データ・ハンドラーが必要です。

デフォルト値は `text/xml` です。

ErrorQueue

処理できなかったメッセージの送信先のキュー。

デフォルト値は `queue://<queue_manager_name>/WC_MQCONN.ERROR` です。

FeedbackCodeMappingMO

InterChange Server にメッセージの受信を同期で肯定応答する場合に使用するデフォルトのフィードバック・コードをオーバーライドし、再割り当てすることができます。このプロパティーでは、各属性名を認識してフィードバック・コードを表すメタオブジェクトを指定できます。フィードバック・コードの対応する値は、InterChange Server に渡される戻り状況です。デフォルトのフィードバック・コードのリストについては、9 ページの『同期の引き渡し』を参照してください。アダプターは、MQ 固有のフィードバック・コードを表す以下の属性値を受け取ります。

- MQFB_APPL_FIRST
- MQFB_APPL_FIRST_OFFSET_N (ここで、N は整数で、MQFB_APPL_FIRST + N の値として解釈されます)

アダプターがメタオブジェクトの属性値として受け入れる ICS 固有の状況コードは、以下のとおりです。

- SUCCESS
- FAIL
- APP_RESPONSE_TIMEOUT
- MULTIPLE_HITS
- UNABLE_TO_LOGIN
- VALCHANGE
- VALDUPES

以下の表に、サンプル・メタオブジェクトを示します。

表 8. サンプルのフィードバック・コード・メタオブジェクト属性

属性名	デフォルト値
MQFB_APPL_FIRST	SUCCESS
MQFB_APPL_FIRST + 1	FAIL
MQFB_APPL_FIRST + 2	UNABLE_TO_LOGIN

デフォルト値は None です。

HostName

WebSphere MQ のサービスを提供するサーバーの名前。

デフォルト値は None です。

InDoubtEvents

予想外のアダプターのシャットダウンにより完全に処理されなかった処理中イベントを処理する方法を指定します。初期化時に処理中キュー内にイベントが見つかった場合取るアクションを、以下の 4 つの中から選択します。

- **FailOnStartup**。エラーをログに記録し、即時にシャットダウンします。
- **Reprocess**。まず残っているイベントを処理してから、入力キュー内にあるメッセージを処理します。
- **Ignore**。処理中キュー内にあるメッセージを廃棄します。
- **LogError**。エラーをログに記録しますが、シャットダウンはしません。

デフォルト値は **Reprocess** です。

InputQueue

アダプターがポーリングして新しいメッセージを探すメッセージ・キュー。複数のキュー名 (セミコロンを区切り文字とする) を指定することができます。例えば、**MyQueueA**、**MyQueueB**、**MyQueueC** の 3 つのキューをポーリングする場合、コネクタ構成プロパティ **InputQueue** の値は **MyQueueA;MyQueueB;MyQueueC** になります。

InputQueue プロパティが指定されていない場合、コネクタは正常に始動しますが、警告メッセージが出力され、要求の処理しか実行されず、イベントの処理は実行されません。

アダプターは、ラウンドロビン方法でキューをポーリングし、各キューから **pollQuantity** の数を上限としてメッセージを取得します。例えば、**pollQuantity** が 2 で、**MyQueueA** に 2 つのメッセージ、**MyQueueB** に 1 つのメッセージ、**MyQueueC** に 5 つのメッセージが含まれている場合、アダプターは以下のようにメッセージを取り出します。

pollQuantity の値は 2 なので、アダプターは **pollForEvents** への呼び出しごとに各キューから最大 2 つのメッセージを取り出します。最初の循環 (2 回のうちの 1 回目) では、アダプターは **MyQueueA**、**MyQueueB**、**MyQueueC** それぞれから先頭のメッセージを取り出します。これで、1 回目のポーリングが完了します。**pollQuantity** の値が 1 の場合は、アダプターはこれで停止します。

pollQuantity の値は 2 なので、アダプターは 2 回目のポーリング (2 回のうちの 2 回目) を開始し、**MyQueueA** と **MyQueueC** からそれぞれ 1 つのメッセージを取り出します。**MyQueueB** は空なので、スキップします。すべてのキューをそれぞれ 2 回ずつポーリングした後、メソッド **pollForEvents** への呼び出しは完了します。メッセージを取り出す順序は以下のようになります。

1. **MyQueueA** から 1 つのメッセージ
2. **MyQueueB** から 1 つのメッセージ

3. MyQueueC から 1 つのメッセージ
4. MyQueueA から 1 つのメッセージ
5. MyQueueB は空なのでスキップする
6. MyQueueC から 1 つのメッセージ

デフォルト値は `queue://<queue_manager_name>/WC_MQCONN.IN` です。

InProgressQueue

処理中にメッセージを保持するメッセージ・キュー。

デフォルト値は `queue://<queue_manager_name>/WC_MQCONN.IN_PROGRESS` です。

PollQuantity

`pollForEvents` スキャン時に `InputQueue` プロパティで指定された各キューから取得するメッセージの数。

デフォルト値は 1 です。

Port

WebSphere MQ リスナーに対して設定されているポート。

デフォルト値は `None` です。

ReplyToQueue

アダプターが要求を発行したときの応答メッセージの引き渡し先のキュー。

デフォルト値は `queue://<queue_manager_name>/WC_MQCONN.REPLY` です。

UnsubscribedQueue

サブスクライブされていないメッセージの送信先のキュー。

デフォルト値は `queue://<queue_manager_name>/WC_MQCONN.UNSUBSCRIBED` です。

注: * 間違っている場合や、認識されていない場合があるので、WebSphere MQ が提供する値は必ず確認してください。問題がある場合は、暗黙的に値を指定してください。

UseDefaults

Create 操作では、`UseDefaults` が `true` に設定されている場合、コネクタは、`isRequired` ビジネス・オブジェクト属性ごとに有効な値またはデフォルト値が設定されているかどうかを検査します。値が設定されている場合、Create 操作は正常に実行されます。パラメーターが `false` に設定されている場合は、コネクタの検査対象が有効な値のみとなるので、有効な値が設定されていないと、Create 操作は異常終了します。デフォルト値は `false` です。

保証付きイベント・デリバリー機能の使用可能化

以下の方法のいずれかで、JMS 対応コネクタに合わせて保証付きイベント・デリバリー機能を構成することができます。

- コネクターが (JMS ソース・キューとして実装された) JMS イベント・ストアを使用する場合、コネクター・フレームワークで JMS イベント・ストアを管理できます。詳細については、『JMS イベント・ストアを使用するコネクターへの保証付きイベント・デリバリー』を参照してください。
- コネクターが JDBC 表、E メール・メールボックス、フラット・ファイルなどの非 JMS イベント・ストアを使用する場合、コネクター・フレームワークは JMS モニター・キューを使用して、重複イベントが発生しないようにすることができます。詳細については、38 ページの『非 JMS イベント・ストアを使用するコネクターへの保証付きイベント・デリバリー』を参照してください。

JMS イベント・ストアを使用するコネクターへの保証付きイベント・デリバリー

JMS 対応コネクターが、そのイベント・ストアを実装するために JMS キューを使用する場合、コネクター・フレームワークは、「コンテナ」として振る舞い、JMS イベント・ストア (JMS ソース・キュー) を管理することができます。単一の JMS トランザクションにおいて、コネクターは、ソース・キューからメッセージを削除し、宛先キューに格納することができます。このセクションでは、JMS イベント・ストアを使用する JMS 対応コネクターの保証付きイベント・デリバリー機能の使用について、以下の情報を提供します。

- 『JMS イベント・ストアを使用するコネクター用の機能の使用可能化』
- 38 ページの『イベント・ポーリングの影響』

JMS イベント・ストアを使用するコネクター用の機能の使用可能化

JMS イベント・ストアを使用する JMS 対応コネクターの保証付きイベント・デリバリー機能を使用可能にするには、コネクター構成プロパティを表 9 に示す値に設定します。

表 9. JMS イベント・ストアを使用するコネクター用の保証付きイベント・デリバリーのコネクター・プロパティ

コネクター・プロパティ	値
DeliveryTransport	JMS
ContainerManagedEvents	JMS
PollQuantity	イベント・ストアの 1 回のポーリングで処理するイベント数。

表9. JMS イベント・ストアを使用するコネクタ用の保証付きイベント・デリバリーのコネクタ・プロパティ (続き)

コネクタ・プロパティ	値
SourceQueue	コネクタ・フレームワークがポーリングし、処理対象のイベントを検索する JMS ソース・キュー (イベント・ストア) の名前。 注: ソース・キューと他の JMS キューは、同じキュー・マネージャーに属している必要があります。コネクタのアプリケーションが、別のキュー・マネージャーに格納されるイベントを生成する場合、リモート・キュー・マネージャーのリモート・キュー定義を指定する必要があります。その後、WebSphere MQ は、リモート・キューから、JMS 対応コネクタが統合ブローカーへの転送に使用するキュー・マネージャーに、イベントを転送することができます。リモート・キュー定義の設定方法の詳細については、IBM WebSphere MQ の資料を参照してください。

コネクタ構成に加えて、JMS ストア内のイベントとビジネス・オブジェクトの間で変換を行うデータ・ハンドラーを構成する必要があります。このデータ・ハンドラー情報は、表 10 に要約されるコネクタ構成プロパティから構成されています。

表 10. 保証付きイベント・デリバリーのデータ・ハンドラー・プロパティ

データ・ハンドラー・プロパティ	値	必須
MimeType	データ・ハンドラーが処理する MIME タイプ。この MIME タイプは、呼び出すデータ・ハンドラーを識別します。	はい
DHClass	データ・ハンドラーを実装する Java クラスの絶対パス名	はい
DataHandlerConfigMOName	MIME タイプとそのデータ・ハンドラーを関連付けるトップレベル・メタオブジェクト名	オプション

注: データ・ハンドラー構成プロパティは、他のコネクタ構成プロパティとともにコネクタ構成ファイルにあります。

JMS イベント・ストアを使用するコネクタを、保証付きイベント・デリバリーを使用するように構成する場合、表 9 および表 10 での説明に従って、コネクタ・プロパティを設定する必要があります。Connector Configurator ツールを使用して、これらのコネクタ構成プロパティを設定します。Connector Configurator は、その「標準のプロパティ」タブに表 9 のコネクタ・プロパティを表示します。Connector Configurator は、「データ・ハンドラー」タブに表 10 のコネクタ・プロパティを表示します。

注: Connector Configurator は、DeliveryTransport コネクタ構成プロパティが JMS、ContainerManagedEvents が JMS に設定されている場合のみ、「データ・ハンドラー」タブのフィールドをアクティブにします。

Connector Configurator の詳細については、79 ページの『付録 B. Connector Configurator』を参照してください。

イベント・ポーリングの影響

ContainedManagedEvents を JMS に設定することにより保証付きイベント・デリバリーを使用しているコネクタは、この機能を持たないコネクタとは若干異なる振る舞いをします。コンテナ管理イベントを提供するために、コネクタ・フレームワークは次の手順でイベント・ストアをポーリングします。

1. JMS トランザクションを始動します。
2. イベント・ストアから JMS メッセージを読み取ります。
イベント・ストアは、JMS ソース・キューとして実装されます。JMS メッセージはイベント・レコードを含んでいます。JMS ソース・キュー名は、SourceQueue コネクタ構成プロパティから取得されます。
3. データ・ハンドラーを呼び出して、イベントをビジネス・オブジェクトに変換します。
コネクタ・フレームワークは、37 ページの表 10 のプロパティを使用して構成されたデータ・ハンドラーを呼び出します。
4. WebSphere MQ Integrator Broker が統合ブローカーの場合、ビジネス・オブジェクトを構成済みワイヤー・フォーマット (XML) に基づいたメッセージに変換します。
5. JMS 宛先キューに、変換されたメッセージを送信します。
WebSphere ICS 統合ブローカーを使用する場合、JMS 宛先キューに送信されるメッセージはビジネス・オブジェクトです。WebSphere MQ Integrator Broker を使用している場合、JMS 宛先キューに送信されるメッセージは (データ・ハンドラーが保証した) XML メッセージです。
6. JMS トランザクションをコミットします。
JMS トランザクションがコミットされると、そのトランザクションにおいて、JMS 宛先キューにメッセージが書き込まれ、JMS ソース・キューから削除されます。
7. ステップ 1 から 6 を繰り返し実行します。PollQuantity コネクタ・プロパティが繰り返しの回数を決定します。

重要: ContainerManagedEvents プロパティを JMS に設定したコネクタは、イベント・ポーリング時に pollForEvents() メソッドを呼び出しません。コネクタの基本クラスに pollForEvents() メソッドが含まれている場合、このメソッドは、呼び出されません。

非 JMS イベント・ストアを使用するコネクタへの保証付きイベント・デリバリー

JMS 対応コネクタが、JDBC イベント表、E メール・メールボックス、フラット・ファイルなどのイベント・ストア実装時に、非 JMS ソリューションを使用する場合、コネクタ・フレームワークは、重複イベント除去を使用して、重複イベントが発生しないようにすることができます。このセクションでは、非 JMS イベント

ント・ストアを使用する JMS 対応コネクタで、保証付きイベント・デリバリー機能の使用について、以下の情報を提供します。

- ・ 『非 JMS イベント・ストアを使用するコネクタでの機能の使用可能化』
- ・ 38 ページの 『イベント・ポーリングの影響』

非 JMS イベント・ストアを使用するコネクタでの機能の使用可能化: 非 JMS イベント・ストアを使用する JMS 対応コネクタで保証付きイベント・デリバリー機能を使用可能にするには、コネクタ構成プロパティを表 11 に示す値に設定します。

表 11. 非 JMS イベント・ストアを使用するコネクタ用の保証付きイベント・デリバリーのコネクタ・プロパティ

コネクタ・プロパティ	値
DeliveryTransport	JMS
DuplicateEventElimination	true
MonitorQueue	コネクタ・フレームワークが、処理済みビジネス・オブジェクトの ObjectEventId を格納する JMS モニター・キューの名前。

保証付きイベント・デリバリーを使用するためにコネクタを構成する場合、表 11 での説明に従って、コネクタ・プロパティを設定する必要があります。

Connector Configurator ツールを使用して、これらのコネクタ構成プロパティを設定します。Connector Configurator ツールは、「標準のプロパティ」タブにコネクタ・プロパティを表示します。Connector Configurator の詳細については、79 ページの 『付録 B. Connector Configurator』を参照してください。

イベント・ポーリングの影響: DuplicateEventElimination を true に設定することにより保証付きイベント・デリバリーを使用しているコネクタは、この機能を持たないコネクタとは若干異なる振る舞いをします。コネクタ・フレームワークは、JMS モニター・キューを使用してビジネス・オブジェクトを追跡し、重複イベント除去を行います。JMS モニター・キュー名は MonitorQueue コネクタ構成プロパティから取得されます。

アプリケーション固有のコンポーネントから (pollForEvents() メソッドの getApplEvent() への呼び出しを通じて) ビジネス・オブジェクトを受信した後、コネクタ・フレームワークは、(getApplEvents() から受信した) 現行のビジネス・オブジェクトが、重複イベントであるかどうか判断する必要があります。この判断を行うために、コネクタ・フレームワークは、JMS モニター・キューからビジネス・オブジェクトを検索し、その ObjectEventId と現行ビジネス・オブジェクトの ObjectEventId とを比較します。

- ・ これら 2 つの ObjectEventId が同じである場合、現行のビジネス・オブジェクトは重複イベントです。この場合、コネクタ・フレームワークは、現行のビジネス・オブジェクトが表すイベントを無視し、このイベントを統合ブローカーに送信しません。
- ・ これら 2 つの ObjectEventIds が同じでない場合、このビジネス・オブジェクトは重複イベントではありません。この場合、コネクタ・フレームワークは、同じ JMS トランザクションにおいて、現行のビジネス・オブジェクトを、JMS モ

ニター・キューにコピーし、JMS デリバリー・キューに配信します。JMS デリバリー・キュー名は `DeliveryQueue` コネクター構成プロパティから取得されず。`gotAppEvent()` メソッドへの呼び出し後、コントロールは、コネクターの `pollForEvents()` に戻ります。

JMS 対応コネクターで重複イベント除去をサポートするためには、コネクターの `pollForEvents()` メソッドに以下のステップが含まれている必要があります。

- 非 JMS イベント・ストアから検索したイベント・レコードからビジネス・オブジェクトを生成する際には、イベント・レコードの固有イベント ID を、そのビジネス・オブジェクトの `ObjectEventId` 属性として保存します。

アプリケーションは、イベント・ストア内のイベント・レコードを一意的に識別するためにイベント ID を生成します。イベントが統合ブローカーに送信されたが、このイベント・レコードの状態が変更可能になる前にコネクターがダウンした場合、そのイベント・レコードは「処理中 (In-Progress)」の状況でイベント・ストアに残ります。コネクターは、バックアップの際、すべての「処理中 (In-Progress)」イベントをリカバリーしなければなりません。コネクターはポーリング再開時に、イベント・ストアに残っているイベント・レコードのビジネス・オブジェクトを生成します。しかし、すでに送信されたビジネス・オブジェクトと新しく生成されたビジネス・オブジェクトは、その `ObjectEventId` として同じイベント・レコードを持つため、コネクター・フレームワークは、新しいビジネス・オブジェクトを重複と判断して、統合ブローカーに送信しない可能性があります。

- コネクターのリカバリー中は、コネクターが新規イベントのポーリングを開始する前に「処理中 (In-Progress)」イベントを処理しなければなりません。コネクターが、始動時に、「処理中 (In-Progress)」イベントを「ポーリング準備 (Ready-for-Poll)」状況に変更しない限り、ポーリング・メソッドは、そのイベント・レコードを再処理しません。

キューの URI (Uniform Resource Identifiers)

キューの URI は、先頭が `queue://` で、その後に以下の要素が続きます。

- キューが常駐するキュー・マネージャーの名前
- スラッシュ /
- キューの名前
- 必要に応じて、その他のキュー・プロパティを設定する `name-value` 形式のリスト

例えば、次の URI を指定した場合、キュー・マネージャー `<queue.manager.name>` に存在するキュー `IN` に接続し、すべてのメッセージが優先順位 5 の WebSphere MQ メッセージとして送信されます。

```
queue://<queue.manager.name>/WC_MQCONN.IN?targetClient=1&priority=5
```

以下の表に、キューの URI のプロパティ名を示します。

表 12. キューの URI に対する WebSphere MQ 固有のコネクター・プロパティ名

プロパティ名	説明	値
expiry	メッセージの存続時間 (ミリ秒単位)。	0 は、無制限です。 正整数は、タイムアウト (ミリ秒単位) を表します。
priority	メッセージの優先順位。	0 から 9 で、1 が最も高い優先順位です。値 -1 は、キューの構成によってプロパティが決まることを意味します。値 -2 は、コネクターが独自のデフォルト値を使用できることを指定します。
persistence	メッセージをディスクに「永久保存」するかどうかを指定します。	1 は非永続です。 2 は永続です。 値 -1 は、キューの構成によってプロパティが決まることを意味します。値 -2 は、コネクターが独自のデフォルト値を使用できることを指定します。
CCSID	宛先の文字セット。	基本 WebSphere MQ マニュアルにリストされている、整数の有効な値です。
targetClient	受信側アプリケーションが JMS に準拠しているかどうかを指定します。	0 は JMS (MQRFH2 ヘッダー) です。 1 は MQ (MQMD ヘッダーのみ) です。
encoding	数値フィールドを表す方法を指定します。	基本 WebSphere MQ マニュアルに記載されている整数値です。

アダプターでは、MQMessages 内のデータの文字セット (CCSID) 属性やエンコード属性を制御することはできません。データ変換は、メッセージ・バッファーとの間でデータをやり取りする際に適用されているので、アダプターは JMS の IBM WebSphere MQ インプリメンテーションに依存してデータを変換します (IBM WebSphere MQ Java クライアント・ライブラリーのマニュアルを参照してください)。したがって、これらの変換は、ネイティブ WebSphere MQ API がオプション MQGMO_CONVERT を使用して実行する変換と双方向で等しくなければなりません。

アダプターは、変換プロセスにおける差異や障害を制御することはできません。アダプターは、特別な変更を必要とせず、WebSphere MQ によってサポートされるすべての CCSID またはエンコードのメッセージ・データを検索できます。特定の CCSID やエンコードのメッセージを転送するには、出力キューが完全修飾 URI であることと、CCSID と encoding の値を指定することが必要です。アダプターはこの情報を WebSphere MQ に渡し、WebSphere MQ は MQMessage のデリバリーのためにデータをエンコードするときに (JMS API を介して) この情報を使用します。

CCSID やエンコードがサポートされていない場合は、通常、IBM の Web サイトから最新バージョンの IBM WebSphere MQ Java クライアント・ライブラリーをダウンロードすることで解決できます。CCSID やエンコードに固有の問題が解決されない場合は、テクニカル・サポートに連絡し、代わりに Java 仮想マシンを使用してアダプターを実行できるかどうかお問い合わせください。

メタオブジェクト属性の構成

Adapter for WebSphere Commerce は、2 種類のメタオブジェクトを認識し、読み取ることができます。

- 静的コネクター・メタオブジェクト
- 動的子メタオブジェクト

動的子メタオブジェクトの属性値は、静的メタオブジェクトの属性値をコピーしてオーバーライドします。

静的メタオブジェクト

WebSphere Commerce の静的メタオブジェクトは、各種ビジネス・オブジェクトで定義されている変換プロパティのリストから構成されています。ビジネス・オブジェクトの変換プロパティを定義するには、まずストリング属性を作成し、構文 `busObj_verb` を使用してその属性に名前を付けます。例えば、動詞 `Create` を使用して `Customer` オブジェクトの変換プロパティを定義するには、`Customer_Create` という名前の属性を作成します。属性のアプリケーション固有のテキストで、実際の変換プロパティを指定します。

さらに、メタオブジェクトでは、`Default` という名前の予約属性を定義することもできます。この属性がある場合は、そのプロパティが、すべてのビジネス・オブジェクト変換プロパティのデフォルト値として使用されます。

注: 静的メタオブジェクトが指定されていない場合、コネクターは、ポーリング時に所定のメッセージ・フォーマットを特定のビジネス・オブジェクト・タイプにマップすることができません。この場合、コネクターは、ビジネス・オブジェクトを指定せずに、メッセージ・テキストを構成済みのデータ・ハンドラーに渡します。データ・ハンドラーがテキストだけに基いてビジネス・オブジェクトを作成できないと、アダプターは、このメッセージ・フォーマットが認識されないことを示すエラーを報告します。

表 13 に、メタオブジェクトのプロパティを示します。

表 13. WebSphere Commerce の静的メタオブジェクトのプロパティ

プロパティ名	説明
<code>CollaborationName</code>	注: これは、同期イベント処理専用のプロパティです (同期イベント処理は、このバージョンのアダプターではサポートされていません)。

表 13. WebSphere Commerce の静的メタオブジェクトのプロパティ (続き)

プロパティ名	説明
DataEncoding	<p>DataEncoding は、メッセージの読み取り/書き込みに使用するエンコードです。このプロパティが静的メタオブジェクトで指定されていない場合、コネクタは特定のエンコードを使用せずにメッセージの読み取りを試みます。動的子メタオブジェクトで定義されている DataEncoding は、静的メタオブジェクトで定義されている値をオーバーライドします。デフォルト値は Text です。この属性の値のフォーマットは、messageType[:enc] です。つまり、有効な値は、</p>
DataHandlerConfigMO	<p>Text:ISO8859_1、Text:UnicodeLittle、Text、Binary です。構成情報を提供するためにデータ・ハンドラーに渡すメタオブジェクト。このプロパティが静的メタオブジェクトで指定されている場合は、DataHandlerConfigMO コネクタ・プロパティで指定されている値がオーバーライドされます。このプロパティが動的子メタオブジェクトで定義されている場合は、コネクタ・プロパティと静的メタオブジェクトのプロパティがオーバーライドされます。指定するビジネス・オブジェクトは、アダプターでサポートされていなければなりません。</p>
DataHandlerMimeType	<p>特定の MIME タイプに基づいてデータ・ハンドラーを指定します。アダプターが WebSphere Commerce とやり取りするためには、MIME タイプとして text/xml を使用しなければなりません。</p>
DoNotReportBusObj	<p>このプロパティが静的メタオブジェクトで指定されている場合は、DataHandlerMimeType コネクタ・プロパティで指定されている値がオーバーライドされます。このプロパティが動的子メタオブジェクトで定義されている場合は、コネクタ・プロパティと静的メタオブジェクトのプロパティがオーバーライドされます。DataHandlerConfigMO で指定するビジネス・オブジェクトは、このプロパティの値に対応する属性を持っていなければなりません。</p> <p>DoNotReportBusObj プロパティは、必要に応じて取り込むことができます。このプロパティを true に設定すると、発行されるすべての PAN レポート・メッセージのメッセージ本体がブランクになります。要求側が、要求が正常に処理されたことは確認したいが、ビジネス・オブジェクトに対する変更の通知は必要としない場合に、このプロパティの設定をお勧めします。これは NAN レポートには影響しません。</p> <p>このプロパティが静的メタオブジェクトで指定されていない場合、アダプターによってデフォルトの false に設定され、メッセージ・レポートにビジネス・オブジェクトが取り込まれます。</p> <p>注: これは、同期イベント処理専用のプロパティです (同期イベント処理は、このバージョンのアダプターではサポートされていません)。</p>

表 13. WebSphere Commerce の静的メタオブジェクトのプロパティ (続き)

プロパティ名	説明
InputFormat	InputFormat は、所定のビジネス・オブジェクトに関連付けるメッセージ・フォーマットです。取得したメッセージがこのフォーマットであると、そのメッセージは可能な限り所定のビジネス・オブジェクトに変換されます。
OutputFormat	OutputFormat は、所定のビジネス・オブジェクトから作成するメッセージに対するプロパティです。OutputFormat が指定されていない場合は、入力フォーマットが使用されます (使用可能な場合)。OutputFormat が動的子メタオブジェクトで定義されている場合、静的メタオブジェクトで定義されている値はオーバーライドされます。
InputQueue	コネクタが新規メッセージを検出するためにポーリングする入力キュー。複数の InputQueues を構成するため、また状況に応じて、別のデータ・ハンドラーを各キューにマップするために、コネクタ固有プロパティを使用することができます。
OutputQueue	OutputQueue は、所定のビジネス・オブジェクトから生成されたメッセージを引き渡す出力キューです。OutputQueue が動的子メタオブジェクトで定義されている場合、静的メタオブジェクトで定義されている値はオーバーライドされます。
ResponseTimeout	応答を待機する場合のタイムアウト時間 (ミリ秒単位) を指定します。この値が定義されていなかったり、値がゼロより小さい場合、アダプターは応答を待機せずに即時に SUCCESS を戻します。ResponseTimeout が動的子メタオブジェクトで定義されている場合、静的メタオブジェクトで定義されている値はオーバーライドされます。
TimeoutFatal	このプロパティが定義され、値が True の場合、ResponseTimeout で指定されている時間内に応答を受け取らないと、アダプターは APP_RESPONSE_TIMEOUT を戻します。応答メッセージを待機しているその他のスレッドはすべて、即時に InterChange Server に APP_RESPONSE_TIMEOUT を戻します。これにより、InterChange Server はアダプターを終了します。TimeoutFatal が動的子メタオブジェクトで定義されている場合、静的メタオブジェクトで定義されている値はオーバーライドされます。

表 14. Customer_Create の場合の JMS の静的メタオブジェクト構造

属性名	アプリケーション固有のテキスト
Customer_Create	DataEncoding=Text:UnicodeLittle; InputFormat=CUST_IN; OutputFormat=CUST_OUT; OutputQueue=QueueA; ResponseTimeout=10000; TimeoutFatal=False

アプリケーション固有のテキスト

アプリケーション固有のテキストは、セミコロンを区切り文字とする name-value 形式で指定します。例えば、次のようになります。

```
InputFormat=CUST_IN;OutputFormat=CUST_OUT
```

データ・ハンドラーから InputQueues へのマッピング

静的メタオブジェクトのアプリケーション固有情報の InputQueue プロパティを使用して、データ・ハンドラーと入力キューを関連付けることができます。この機能は、それぞれが異なるフォーマットと変換要件を持つ複数の取り引き先パートナーを処理する際に便利です。これを行うには、以下の手順を実行する必要があります。

1. コネクター固有のプロパティ (34 ページの『InputQueue』参照) を使用して、1 つ以上の入力キューを構成します。
2. 各入力キューにキュー・マネージャーを指定し、アプリケーション固有の情報に、データ・ハンドラー・クラス名や MIME タイプとともに入力キュー名を指定します。

例えば、静的メタオブジェクト内の以下の属性は、データ・ハンドラーと CompReceipts という名前の InputQueue を関連付けます。

```
[Attribute]
Name = Cust_Create
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = InputQueue=//queue.manager/CompReceipts;
  DataHandlerClassName=com.crossworlds.
  DataHandlers.MQ.disposition_notification;DataHandlerMimeType=message/
  disposition_notification
IsRequiredServerBound = false
[End]
```

入力フォーマットの多重定義

メッセージを取得する場合、コネクターは通常、入力フォーマットを、ビジネス・オブジェクトと動詞の組み合わせに固有のフォーマットと突き合わせます。そして、アダプターは、ビジネス・オブジェクトの名前とメッセージの内容をデータ・ハンドラーに渡します。これで、データ・ハンドラーは、メッセージの内容が、ユーザーが必要とするビジネス・オブジェクトに対応するかどうかを検証することができます。

ただし、複数のビジネス・オブジェクトで同じ入力フォーマットが定義されていると、コネクターは、データ・ハンドラーにデータを渡す前に、そのデータがどのビジネス・オブジェクトを表しているか判別できません。このような場合、アダプターはデータ・ハンドラーにメッセージの内容だけを渡してから、生成されるビジネス・オブジェクトに基づいて変換プロパティを探します。つまり、データ・ハンドラーは、メッセージの内容だけに基づいてビジネス・オブジェクトを判別しなければなりません。

生成されたビジネス・オブジェクトで動詞が設定されていない場合は、コネクターは、任意の動詞を使って、このビジネス・オブジェクトに対して定義されている変換プロパティを探します。見つかった変換プロパティ・セットが 1 つだけの場合は、コネクターは、その指定した動詞を割り当てます。複数のプロパティが見つかった場合は、コネクターは動詞を見分けることができないので、メッセージを異常終了します。

サンプル・メタオブジェクト

以下に示すメタオブジェクトでは、動詞 Create を使用して Customer ビジネス・オブジェクトを変換するようにコネクタが構成されます。

```
[BusinessObjectDefinition]
Name = MO_WebSphereCommerceConfig
Version = 3.0.0

  [Attribute]
  Name = Default
  Type = String
  MaxLength = 1
  IsKey = true
  IsForeignKey = false
  IsRequired = false
  AppSpecificInfo = OutputQueue=queue://<Queue Manager
    Name>/WCS_Serial_Inbound?targetClient=1;
    OutputFormat=MQSTR
  IsRequiredServerBound = false
  [End]
  [Attribute]
  Name = WCS_Create_WCS_Customer_Create
  Type = String
  MaxLength = 255
  IsKey = false
  IsForeignKey = false
  IsRequired = false
  AppSpecificInfo = OutputQueue=queue://<Queue Manager
    Name>/WCS_Serial_Inbound?targetClient=1;
    OutputFormat=MQSTR
  IsRequiredServerBound = false
  [End]
  [Attribute]
  Name = WCS_Report_NC_PurchaseOrder_Create
  Type = String
  MaxLength = 255
  IsKey = false
  IsForeignKey = false
  IsRequired = false
  AppSpecificInfo = InputFormat=MQSTR
  IsRequiredServerBound = false
  [End]
  [Attribute]
  Name = ObjectEventId
  Type = String
  MaxLength = 255
  IsKey = false
  IsForeignKey = false
  IsRequired = false
  IsRequiredServerBound = false
  [End]

  [Verb]
  Name = Create
  [End]

  [Verb]
  Name = Delete
  [End]

  [Verb]
  Name = Retrieve
  [End]
```

```
[Verb]
Name = Update
[End]
[End]
```

動的子メタオブジェクト

静的メタオブジェクトからメタデータを指定できない場合、コネクタは、ビジネス・オブジェクトのインスタンスごとにランタイムで指定されるメタデータを受け入れることができます。

コネクタは、コネクタに渡されるトップレベルのビジネス・オブジェクトに子として追加されている動的メタオブジェクトから、変換プロパティを認識して読み取ります。コネクタを構成する場合に使用する静的メタオブジェクトで指定できる変換プロパティは、動的子メタオブジェクトの属性値にコピーされます。

動的子メタオブジェクトのプロパティは、静的メタオブジェクトのプロパティをオーバーライドするので、動的子メタオブジェクトを指定する場合は、静的メタオブジェクトを指定するコネクタ・プロパティを取り込む必要はありません。したがって、動的子メタオブジェクトは、静的メタオブジェクトとは切り離して使用することができます (静的メタオブジェクトを動的子メタオブジェクトと切り離して使用することもできます)。

表 14 および 表 15 に、ビジネス・オブジェクト `Customer_Create` の静的メタオブジェクトと動的子メタオブジェクトのサンプルを示します。アプリケーション固有のテキストは、セミコロンを区切り文字とする `name-value` の形式で構成されることに注意してください。

表 15. `Customer_Create` の場合の `WebSphere Commerce` の動的子メタオブジェクト構造

属性名	値
<code>DataEncoding</code>	<code>Text:UnicodeLittle</code>
<code>DataHandlerMimeType^a</code>	<code>text/delimited</code>
<code>OutputFormat</code>	<code>CUST_OUT</code>
<code>OutputQueue</code>	<code>QueueA</code>
<code>ResponseTimeout</code>	<code>10000</code>
<code>TimeoutFatal</code>	<code>False</code>

a. `DataHandlerConfigMO` は、コネクタ構成プロパティまたは静的メタオブジェクトのいずれかで指定されているものと見なします。

コネクタは、トップレベルのビジネス・オブジェクトのアプリケーション固有のテキストをチェックして、タグ `cw_mo_conn` が子メタオブジェクトを指定しているかどうかを判別します。指定している場合、動的子メタオブジェクト値は、静的メタオブジェクトで指定されている値をオーバーライドします。

ポーリング時の活動

ポーリング時に取得したメッセージの詳細情報をコラボレーションに提供するために、コネクタは動的メタオブジェクトの特定の属性を生成します (作成したビジネス・オブジェクトですでに定義されている場合)。

表 16 に、ポーリング時の動的子メタオブジェクトの構造を示します。

表 16. ポーリング時の JMS の動的子メタオブジェクトの構造

属性名	サンプル値
InputFormat	CUST_IN
InputQueue	MYInputQueue
OutputFormat	CxIgnore
OutputQueue	CxIgnore
ResponseTimeout	CxIgnore
TimeoutFatal	CxIgnore

表 16 に示すように、動的子メタオブジェクトでは、追加属性 `InputQueue` を定義することができます。この属性には、所定のメッセージの取得先のキューの名前が入ります。このプロパティは、子メタオブジェクトで定義されていない場合は生成されません。

シナリオ例:

- コネクターは、キュー `MyInputQueue` から、フォーマットが `CUST_IN` のメッセージを取得します。
- コネクターは、このメッセージを `Customer` ビジネス・オブジェクトに変換し、アプリケーション固有のテキストをチェックして、メタオブジェクトが定義されているかどうか判別します。
- メタオブジェクトが定義されている場合、コネクターはこのメタオブジェクトのインスタンスを作成し、それに応じて `InputQueue` 属性と `InputFormat` 属性を生成して、ビジネス・オブジェクトを利用可能なコラボレーションにパブリッシュします。

動的子メタオブジェクトのサンプル

```
[BusinessObjectDefinition]
Name = MO_Sample_Config
Version = 1.0.0

  [Attribute]
  Name = OutputFormat
  Type = String
  MaxLength = 1
  IsKey = true
  IsForeignKey = false
  IsRequired = false
  DefaultValue = CUST
  IsRequiredServerBound = false
  [End]
  [Attribute]
  Name = OutputQueue
  Type = String
  MaxLength = 1
  IsKey = false
  IsForeignKey = false
  IsRequired = false
  DefaultValue = OUT
  IsRequiredServerBound = false
  [End]
  [Attribute]
  Name = ResponseTimeout
  Type = String
  MaxLength = 1
```

```

IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = -1
IsRequiredServerBound = false
[End]
[Attribute]
Name = TimeoutFatal
Type = String
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = InputFormat
Type = String
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = InputQueue
Type = String
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

[Verb]
Name = Update
[End]
[End]
[BusinessObjectDefinition]
Name = Customer
Version = 1.0.0
AppSpecificInfo = cw_mo_conn=MyConfig

[Attribute]
Name = FirstName

```

```

Type = String
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = LastName
Type = String
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = Telephone
Type = String
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = MyConfig
Type = MO_Sample_Config
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

[Verb]
Name = Update
[End]
[End]

```

アダプターの複数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、51 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. 始動スクリプトのショートカットを作成します (Windows のみ)。
4. 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。

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

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

始動ファイルの構成

Adapter for WebSphere Commerce を始動するには、始動ファイルを構成しなければなりません。

Windows

Windows プラットフォーム用にアダプターの構成を実行する場合は、始動ファイル (アダプターに付属の、`start_WebSphereCommerceAdapter.bat` または `start_WebSphereCommerce.bat` のいずれかのファイル) を変更する必要があります。

1. `start_WebSphereCommerceAdapter.bat` ファイルを開きます。
2. “Set the directory containing your WebSphere MQ Java client libraries” から始まるセクションまでスクロールし、WebSphere MQ Java クライアント・ライブラリーのロケーションを指定します。

UNIX

UNIX プラットフォーム用にアダプターの構成を実行する場合は、始動ファイル (アダプターに付属の、`start_WebSphereCommerceAdapter.sh` または `start_WebSphereCommerce.sh` のいずれかのファイル) を変更する必要があります。

1. `start_WebSphereCommerceAdapter.sh` ファイルを開きます。
2. “Set the directory containing your WebSphere MQ Java client libraries” から始まるセクションまでスクロールし、WebSphere MQ Java クライアント・ライブラリーのロケーションを指定します。

コネクタの開始

コネクタは、**コネクタ始動スクリプト**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のランタイム・ディレクトリに存在していなければなりません。

`ProductDir\connectors\connName`

ここで、`connName` はコネクタを示します。始動スクリプトの名前は、表 17 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 17. コネクタの始動スクリプト

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	<code>connector_manager_connName</code>
Windows	<code>start_connName.bat</code>

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

- Windows システムで「スタート」メニューから。
「プログラム」>「IBM WebSphere Business Integration Adapters」>「アダプター」>「コネクタ」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 [-cconfigFile]`
 - UNIX ベースのシステム:
`connector_manager_connName -start`

ここで、`connName` はコネクタの名前であり、`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文字列を指定します。

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、コネクタのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。

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

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

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

コネクターの停止

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

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

```
connector_manager_connName -stop
```

ここで、*connName* はコネクターの名前です。

- Adapter Monitor から (WebSphere Business Integration Adapters 製品のみ)。Adapter Monitor は System Manager 始動時に起動されます。このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。
- System Monitor から (WebSphere InterChange Server 製品のみ) このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。
- Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムのシャットダウン時に、コネクターは停止します。

第 3 章 ビジネス・オブジェクトの処理

この章では、アダプターがビジネス・オブジェクトを処理する方法と、コネクタに存在する前提事項について説明します。この情報を参考に、新規ビジネス・オブジェクトを実装してください。

この章ではまた、アダプターがビジネス・オブジェクト処理する方法と、コネクタの前提事項についても説明します。この情報を参考に、新規ビジネス・オブジェクトを実装してください。この章は、以下のトピックから構成されます。

- 『ビジネス・オブジェクトの構造』
- 56 ページの『エラー処理』
- 57 ページの『トレース』

アダプターには、サンプルのビジネス・オブジェクトしか添付されていません。システム・インテグレーター、コンサルタントまたは顧客は、特定のビジネス・オブジェクトを作成する必要があります。

サンプルのビジネス・オブジェクトは、アダプター・パッケージの /samples ディレクトリに入っています。

アダプターはメタデータ主導型です。ここで、メタデータとはアプリケーションに関するデータのことです。このデータはビジネス・オブジェクト定義に格納され、アダプターとアプリケーションとのやり取りに役立ちます。メタデータ主導型コネクタは、サポートする各ビジネス・オブジェクトを処理する際に、コネクタ内にハードコーディングされた命令ではなく、ビジネス・オブジェクト定義にエンコードされたメタデータに基づいて処理を行います。

ビジネス・オブジェクトのメタデータには、ビジネス・オブジェクトの構造、属性プロパティの設定、およびアプリケーション固有テキストの内容が含まれています。コネクタはメタデータ主導型であるため、アダプター・コードを変更する必要なしに、新規または変更されたビジネス・オブジェクトを処理することができます。ただし、アダプターの構成済みデータ・ハンドラーでは、そのビジネス・オブジェクトの構造、オブジェクトの基数、アプリケーション固有のテキストのフォーマット、およびビジネス・オブジェクトのデータベース表現について前提事項が存在します。そのため、アダプターのビジネス・オブジェクトを作成または変更する場合、その変更内容は、アダプターが従うべきルールに準拠している必要があります。これに準拠していないと、コネクタは新規または変更されたビジネス・オブジェクトを正しく処理できません。

ビジネス・オブジェクトの構造

アダプターをインストールしたら、ビジネス・オブジェクトを作成する必要があります。

Adapter for WebSphere Commerce は、キューから WebSphere MQ メッセージを取得し、そのメッセージに含まれているビジネス・データを (メタオブジェクトで定義されている) ビジネス・オブジェクトに取り込むを試みます。

アダプターが交換する MQ Series メッセージ内のビジネス・データは、XML 文書に収められています。アダプターは、XML データ・ハンドラーを使用して、XML 文書からビジネス・オブジェクト、およびビジネス・オブジェクトから XML 文書にデータを変換します。

ビジネス・オブジェクト定義の構造は、XML データ・ハンドラーの要件に準拠していなければなりません。これらの要件、および XML DTD と Edifecs SpecBuilder ユーティリティを使用して DTD をビジネス・オブジェクト定義に変換する方法については、IBM WebSphere Business Integration の「*Data Handler for XML*」を参照してください。

WebSphere Commerce の一般的な Customer_Create ビジネス・オブジェクトで使用するプロパティと構造を表示するには、`file/connector/WebSphereCommerce/samples/WC_BODefinition.in` を開きます。

エラー処理

アダプターで生成されたエラー・メッセージはすべて、`WebSphereCommerce.txt` というメッセージ・ファイルに保管されます。(このファイルの名前は、`LogFileName` 標準コネクタ構成プロパティで指定します。)各エラーにはエラー番号が割り振られ、その後にエラー・メッセージが挿入されます。

Message number
Message text

以下のセクションで、アダプターが処理する特定のエラーについて説明します。

アプリケーション・タイムアウト

エラー・メッセージ `BON_APPRESPONSETIMEOUT` が戻されるのは、以下の場合です。

- アダプターが、メッセージの取得時に WebSphere MQ への接続を確立できなかった場合。
- コネクタが、ビジネス・オブジェクトをメッセージに正常に変換したが、接続障害のため、宛先のキューにそのメッセージを送信できなかった場合。
- アダプターが、メッセージを発行したが、変換プロパティ `TimeoutFatal` が `True` に設定されているビジネス・オブジェクトについて、応答待機時間がタイムアウトした場合。
- アダプターが、戻りコードが `APP_RESPONSE_TIMEOUT` または `UNABLE_TO_LOGIN` に設定されている応答メッセージを受け取った場合。

アンサブスクライブされたビジネス・オブジェクト

アダプターは、アンサブスクライブされたビジネス・オブジェクトに関するメッセージを取得すると、`UnsubscribedQueue` プロパティで指定されているキューに引き渡します。

注: UnsubscribedQueue が定義されていない場合、アンサブスクライブされたメッセージは廃棄されます。

データ・ハンドラーによる変換

データ・ハンドラーがメッセージをビジネス・オブジェクトに変換できなかった場合、またはビジネス・オブジェクトに固有の (WebSphere MQ には関係ない) 処理エラーが発生した場合、メッセージは ErrorQueue で指定されたキューに引き渡されます。ErrorQueue が定義されていない場合、エラーが原因で処理できなかったメッセージは廃棄されます。

データ・ハンドラーがビジネス・オブジェクトをメッセージに変換できなかった場合は、BON_FAIL が戻されます。

トレース

トレースは、アダプターの振る舞いを詳しく追跡できる、オプションのデバッグ機能です。デフォルトでは、トレース・メッセージは STDOUT に書き込まれます。トレース・メッセージの構成については、コネクタ構成プロパティを参照してください。トレースの使用可能化と設定の方法など、トレースの詳細については、「コネクタ開発ガイド」を参照してください。

以下に、トレース・メッセージの推奨コンテンツを示します。

- レベル 0 トレース・メッセージでアダプター・バージョンを識別する場合は、このレベルを使用します。
- レベル 1 トレース・メッセージで、処理されるビジネス・オブジェクトごとに主要な情報を提供する、あるいはポーリング・スレッドが入力キューで新しいメッセージを検出するごとに記録する場合は、このレベルを使用します。
- レベル 2 ビジネス・オブジェクトが `gotApp1Event()` または `executeCollaboration()` のいずれかから InterChange Server に通知されるごとにトレース・メッセージをログに記録する場合は、このレベルを使用します。
- レベル 3 トレース・メッセージで、メッセージからビジネス・オブジェクトへの変換およびビジネス・オブジェクトからメッセージへの変換に関する情報を提供したり、出力キューへのメッセージの転送に関する情報を提供したりする場合は、このレベルを使用します。
- レベル 4 トレース・メッセージで、アダプターが機能の実行を開始または停止したときを識別する場合は、このレベルを使用します。
- レベル 5 トレース・メッセージで、アダプターの初期化や、アプリケーションで実行されたステートメント、メッセージがキューから取り出された時間やキューに入った時間を示したり、ビジネス・オブジェクトのダンプを記録したりする場合は、このレベルを使用します。

第 4 章 トラブルシューティング

この章では、アダプターを始動または実行するときに発生する可能性がある問題について説明します。

始動時の問題

問題

初期化中にアダプターが予期しないエラーでシャットダウンし、次のメッセージが報告されました。

```
Exception in thread "main"  
java.lang.NoClassDefFoundError:  
javax/jms/JMSException...
```

初期化中にアダプターが予期しないエラーでシャットダウンし、次のメッセージが報告されました。

```
Exception in thread "main"  
java.lang.NoClassDefFoundError:  
com/ibm/mq/jms/MQConnectionFactory...
```

初期化中にアダプターが予期しないエラーでシャットダウンし、次のメッセージが報告されました。

```
Exception in thread "main"  
java.lang.NoClassDefFoundError:  
javax/naming/Referenceable...
```

初期化中にアダプターが予期しないエラーでシャットダウンし、次のメッセージが報告されました。

```
java.lang.UnsatisfiedLinkError: no mqjbnd01 in  
shared library path
```

アダプターから次のメッセージが報告されました。

```
MQJMS2005: failed to create MQQueueManager for ':'  
アダプターから、クラスが見つからなかったことが報告  
されました。
```

アダプターが始動時にハングします。

考えられる処置/説明

アダプターが IBM WebSphere MQ Java クライアント・ライブラリーでファイル `.jms.jar` を見つけることができません。 `start_connector.bat` 内の変数

`MQSERIES_JAVA_LIB` が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認します。

アダプターが IBM WebSphere MQ Java クライアント・ライブラリーでファイル `com.ibm.mqjms.jar` を見つけることができません。 `start_connector.bat` 内の変数

`MQSERIES_JAVA_LIB` が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認します。

アダプターが IBM WebSphere MQ Java クライアント・ライブラリーでファイル `jndi.jar` を見つけることができません。 `start_connector.bat` 内の変数

`MQSERIES_JAVA_LIB` が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認します。

アダプターが IBM WebSphere MQ Java クライアント・ライブラリーで必要なランタイム・ライブラリー `mqjbnd01.dll` [NT] または `libmqjbnd01.so` [Solaris]) を見つけることができません。パスに IBM WebSphere MQ Java クライアント・ライブラリーのフォルダーが含まれていることを確認します。

プロパティー `HostName`、`Channel`、および `Port` の値を明示的に設定します。

クラスのパスに以下のディレクトリーが含まれていることを検証します。

```
<MQSeries Install>%java]lib
```

```
<install_path>%bin
```

WebSphere MQ リスナーが実行されていることを検証します。

イベント処理

問題

アダプターは、MQRFH2 ヘッダーを持つすべてのメッセージをデリバリーします。

アダプターは、アダプターのメタオブジェクト内のメッセージ・フォーマットの定義にかかわらず、デリバリー時にすべてのメッセージ・フォーマットの 8 文字を超える部分を切り捨てます。

考えられる処置/説明

MQMD WebSphere MQ ヘッダーを持つメッセージのみをデリバリーするには、出力キューの URI の名前に `?targetClient=1` を付加します。例えば、メッセージをキュー `queue://my.queue.manager/OUT` に出力する場合は、URI を `queue://my.queue.manager/OUT?targetClient=1` に変更します。詳細については、17 ページの『第 2 章コネクタのインストールおよび構成』を参照してください。

これは、WebSphere MQ MQMD メッセージ・ヘッダーの制限であり、アダプターの制限ではありません。

付録 A. コネクターの標準構成プロパティ

この付録では、WebSphere Business Integration アダプターのコネクタ・コンポーネントの標準構成プロパティ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)

コネクタによっては、一部の標準プロパティ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がある標準プロパティのリストが表示されます。

コネクタ固有のプロパティの詳細については、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・パスの区切り記号として円記号 (¥) 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。

新規プロパティと削除されたプロパティ

本リリースには、次の標準プロパティが追加されました。

新規プロパティ

- XMLNamespaceFormat

削除されたプロパティ

- RestartCount

標準コネクタ・プロパティの構成

アダプター・コネクタには 2 つのタイプの構成プロパティがあります。

- 標準構成プロパティ
- コネクタ固有のプロパティ

このセクションでは、標準構成プロパティについて説明します。コネクタ固有の構成プロパティについては、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・プロパティを構成します。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、79 ページの『付録 B. Connector Configurator』を参照してください。

注: Connector Configurator と System Manager は、Windows システム上でのみ動作します。コネクタを UNIX システム上で稼働している場合でも、これらのツールがインストールされた Windows マシンが必要です。UNIX 上で動作するコネクタのコネクタ・プロパティを設定する場合は、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、コネクタ用の Connector Configurator を開く必要があります。

プロパティ値の設定と更新

プロパティ・フィールドのデフォルトの長さは 255 文字です。

コネクタは、以下の順序に従ってプロパティの値を決定します (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 (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・プロパティの値を変更する場合は、プロパティの**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・プロパティには、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 で稼働している場合など、コネクタがスタンドアロン・モードで (System Manager から独立して) 稼働している場合は、構成ファイルでのみプロパティを変更できます。この場合、動的更新は実行できません。

• コンポーネント再始動

System Manager でコネクタを停止してから再始動しなければ、変更が有効になりません。アプリケーション固有コンポーネントまたは統合ブローカーを停止、再始動する必要はありません。

• サーバー再始動

アプリケーション固有のコンポーネントおよび統合ブローカーを停止して再始動しなければ、変更が有効になりません。

• エージェント再始動 (ICS のみ)

アプリケーション固有のコンポーネントを停止して再始動しなければ、変更が有効になりません。

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

標準プロパティの要約

表 18 は、標準コネクタ構成プロパティの早見表です。標準プロパティの依存関係は RepositoryDirectory に基づいているため、コネクタによっては使用されないプロパティがあり、使用する統合ブローカーによってプロパティの設定が異なる可能性があります。

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

表 18. 標準構成プロパティの要約

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

表 18. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
DuplicateEventElimination	True または False	False	コンポーネント再始動	JMS トランスポートのみ: Container Managed Events は <NONE> でなければならぬ
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 の場合は localhost:2506 を使用。	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMaxNativeStackSize	スタックのサイズ (キロバイト単位)	128k	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	Repository Directory は <REMOTE>
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならぬ
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> でなければならぬ

表 18. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならぬ
MessageFileName	パスまたはファイル名	InterchangeSystem.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	CONNECTORNAME/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は True でなければならぬ
OADAutoRestartAgent	True または False	False	動的	Repository Directory は <REMOTE> でなければならぬ
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければならぬ
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければならぬ
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 18. 標準構成プロパティの要約 (続き)

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

このセクションでは、各標準コネクタ構成プロパティの定義を示します。

AdminInQueue

統合ブローカーからコネクタ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

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

デフォルト値は `CONNECTORNAME/ADMINOUTQUEUE` です。

AgentConnections

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

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

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

AgentTraceLevel

アプリケーション固有のコンポーネントのトレース・メッセージのレベルです。デフォルト値は 0 です。コネクタは、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。コネクタを実行する前に、このプロパティ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 ベースのコネクタでは、このプロパティは使用しません。C++ コネクタでは、現在、このプロパティに `ascii7` という値が使用されています。

デフォルトでは、ドロップ・リストには、サポートされる文字エンコードの一部のみが表示されます。ドロップ・リストに、サポートされる他の値を追加するには、製品ディレクトリーにある `¥Data¥Std¥stdConnProps.xml` ファイルを手動で変更する必要があります。詳細については、79 ページの『付録 B. Connector Configurator』を参照してください。

ConcurrentEventTriggeredFlows

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

コネクタ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・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` 構成プロパティに、1 より大きい値を設定します。

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

ContainerManagedEvents

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

デフォルト値は `No value` です。

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

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

また、MimeType、DHClass、および DataHandlerConfigMOName (オプション) プロパティを設定したデータ・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の値を設定するには、Connector Configurator の「データ・ハンドラー」タブを使用します。「データ・ハンドラー」タブの値のフィールドは、ContainerManagedEvents を JMS に設定した場合にのみ表示されます。

注: ContainerManagedEvents を JMS に設定した場合、コネクタはその 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。

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

ControllerStoreAndForwardMode

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

宛先側のアプリケーション固有のコンポーネントが使用不可であることをコネクタ・コントローラーが検出した場合に、コネクタ・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・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・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・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・コントローラーはその要求を失敗させます。

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

デフォルト値は true です。

ControllerTraceLevel

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

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

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

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

- ICS がブローカー・タイプの場合は、DeliveryTransport プロパティの指定可能な値は 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 プロパティに指定されている値が、MQ または IDL である場合、コネクタは、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・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・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・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、コネクタとクライアント・コネクタ・フレームワークとの間の通信を可能にします。

JMS をデリバリー・トランスポートとして選択した場合は、`jms.MessageBrokerName`、`jms.FactoryClassName`、`jms.Password`、`jms.UserName` などの追加の JMS プロパティが 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です。

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

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

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

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

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

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

この行は、ヒープ・メモリーの使用量を最大 768 MB (3 セグメント * 256 MB) に制限します。プロセス・メモリーがこの制限値を超えると、ページ・スワッピングが発生し、システムのパフォーマンスに悪影響を与える場合があります。

- IPCCBaseAddress プロパティーの値を 11 または 12 に設定する。このプロパティーの詳細については、「システム・インストール・ガイド (UNIX 版)」を参照してください。

DuplicateEventElimination

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

このプロパティーは、false に設定することもできます。

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

FaultQueue

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

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

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

デフォルト値は 128M です。

JvmMaxNativeStackSize

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

デフォルト値は 128K です。

JvmMinHeapSize

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

デフォルト値は 1M です。

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

デフォルト値は crossworlds.queue.manager です。

jms.NumConcurrentRequests

コネクタ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。

このプロパティは、MQ トランSPORTを使用するコネクタにのみ適用されます。DeliveryTransport プロパティには 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、次の形式で指定します。

`ll_TT.codeset`

ここで、以下のように説明されます。

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

デフォルトでは、ドロップ・リストには、サポートされるロケールの一部のみが表示されます。ドロップ・リストに、サポートされる他の値を追加するには、製品ディレクトリーにある `¥Data¥Std¥stdConnProps.xml` ファイルを手動で変更する必要があります。詳細については、79 ページの『付録 B. Connector Configurator』を参照してください。

デフォルト値は `en_US` です。コネクタがグローバル化に対応していない場合、このプロパティの有効な値は `en_US` のみです。特定のコネクタがグローバル化に対応しているかどうかを判断するには、以下の Web サイトにあるコネクタのバージョン・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/wicsserver/infocenter>

LogAtInterchangeEnd

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

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

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

MaxEventCapacity

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

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

MessageFileName

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

コネクタ・メッセージ・ファイルが存在しない場合は、コネクタは 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: 特定のコネクタについて、コネクタ独自のメッセージ・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ズ・ガイドを参照してください。

MonitorQueue

コネクタ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。このプロパティは、DeliveryTransport プロパティ値が 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。

コネクタが自動再始動機能およびリモート再始動機能を使用するかどうかを指定します。この機能では、MQ により起動される Object Activation Daemon (OAD) を使用して、異常シャットダウン後にコネクタを再始動したり、System Monitor からリモート・コネクタ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、このプロパティを true に設定する必要があります。MQ により起動される OAD 機能の構成方法については、「システム・インストール・ガイド (Windows 版)」または「システム・インストール・ガイド (UNIX 版)」を参照してください。

デフォルト値は false です。

OADMaxNumRetry

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

異常シャットダウンの後で MQ により起動される OAD がコネクタ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。このプロパティを有効にするには、OADAutoRestartAgent プロパティを true に設定する必要があります。

デフォルト値は 1000 です。

OADRetryTimeInterval

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

MQ により起動される OAD の再試行時間間隔の分数を指定します。コネクタ・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・コントローラーはコネクタ・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 プロパティで指定された回数だけ繰り返します。このプロパティを有効にするには、OADAutoRestartAgent プロパティを true に設定する必要があります。

デフォルト値は 10 です。

PollEndTime

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

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

PollFrequency

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

- ポーリング・アクション間のミリ秒数。
- ワード key。コネクタは、コネクタのコマンド・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。コネクタはポーリングを実行しません。このワードは小文字で入力します。

デフォルト値は 10000 です。

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

PollQuantity

コネクタ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固有のポーリング数設定プロパティがある場合、標準プロパティの値は、このコネクタ固有のプロパティの設定値によりオーバーライドされます。

PollStartTime

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

このプロパティ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 リポジトリからこの情報を取得するためです。

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

ResponseQueue

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

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

RestartRetryCount

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

デフォルト値は 3 です。

RestartRetryInterval

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

デフォルト値は 1 です。

RHF2MessageDomain

WebSphere Message Brokers および WAS でのみ使用されます。

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

サンプル・ヘッダーを以下に示します。

```
<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、このプロパティには `xml` も設定できます。このプロパティは、`DeliveryTransport` が JMS に設定されており、かつ `WireFormat` が `CwXML` に設定されている場合にだけ表示されます。

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・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、68 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・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・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。

デフォルト値は `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・フレームワークに配信します。このキューは、コネクタが同期実行を使用する場合にのみ必要です。

デフォルト値は `CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE` です。

SynchronousRequestTimeout

`DeliveryTransport` が JMS の場合のみ適用されます。

コネクタ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。コネクタは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージをエラー・メッセージとともに障害キューに移動します。

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

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

WsifSynchronousRequest Timeout

WAS 統合ブローカーでのみ使用されます。

コネクタ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。

- コネクタを構成するためのコネクタ固有のプロパティ・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を設定する

注:

本書では、ディレクトリー・パスの区切り記号として円記号 (¥) 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。

- 79 ページの『Connector Configurator の概要』
- 80 ページの『Connector Configurator の始動』
- 81 ページの『コネクタ固有のプロパティ・テンプレートの作成』
- 84 ページの『新規構成ファイルの作成』
- 87 ページの『構成ファイル・プロパティの設定』
- 95 ページの『グローバル化環境における Connector Configurator の使用』

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・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。

- コネクタを構成するためのコネクタ固有のプロパティ・テンプレートを作成します。
- **コネクタ構成ファイル**を作成します。インストールするコネクタ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を設定します。
場合によっては、コネクタ・テンプレートでプロパティに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・

オブジェクト定義と、ICS の場合はコラボレーションとともに使用するマップを指定し、必要に応じてメッセージング、ロギング、トレース、およびデータ・ハンドラー・パラメーターを指定する必要があります。

Connector Configurator の実行モードと使用する構成ファイルのタイプは、実行する統合ブローカーによって異なります。例えば、使用している統合ブローカーが WMQI の場合、Connector Configurator を System Manager から実行するのではなく、直接実行します (80 ページの『スタンドアロン・モードでの Configurator の実行』を参照)。

コネクタ構成プロパティには、標準の構成プロパティ (すべてのコネクタがもつプロパティ) と、コネクタ固有のプロパティ (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で必要なプロパティ) とが含まれます。

標準プロパティはすべてのコネクタにより使用されるので、標準プロパティ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の値を設定する必要があります。

標準プロパティの範囲は、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があります。Connector Configurator の「標準のプロパティ」ウィンドウには、特定の構成で設定可能なプロパティが表示されます。

ただし**コネクタ固有プロパティ**の場合は、最初にプロパティを定義し、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固有プロパティ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、82 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、コネクタ構成ファイルを編集できます。

これを行うには、以下のステップを実行します。

- 「スタート」>「プログラム」から、「IBM WebSphere InterChange Server」>「IBM WebSphere Business Integration Toolset」>「開発」>「Connector Configurator」をクリックします。

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

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (86 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

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

コネクター固有のプロパティ・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のテンプレートとシステム提供の標準プロパティが必要です。

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

- テンプレートの新規作成については、82 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、新しい名前でのこのテンプレートを保管します。

新規テンプレートの作成

このセクションでは、テンプレートでプロパティを作成し、プロパティの一般特性および値を定義し、プロパティ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、新規コネクタ構成ファイルを作成するためのベースとして使用します。

テンプレートは以下のように作成します。

1. 「ファイル」>「新規」>「コネクタ固有プロパティ・テンプレート」をクリックします。
2. 以下のフィールドを含む「コネクタ固有プロパティ・テンプレート」ダイアログ・ボックスが表示されます。
 - 「テンプレート」、「名前」
このテンプレートが使用されるコネクタ（またはコネクタのタイプ）を表す固有の名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
 - 「旧テンプレート」、「変更する既存のテンプレートを選択してください」
「テンプレート名」表示に、現在使用可能なすべてのテンプレートの名前が表示されます。
 - テンプレートに含まれているコネクタ固有のプロパティ定義を調べるには、「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定義のリストが「テンプレートのプレビュー」表示に表示されます。テンプレートを作成するときには、ご使用のコネクタに必要なプロパティ定義に類似したプロパティ定義が含まれている既存のテンプレートを使用できます。
3. 「テンプレート名」表示からテンプレートを選択し、その名前を「名前の検索」フィールドに入力し（または「テンプレート名」で自分の選択項目を強調表示し）、「次へ」をクリックします。

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

一般特性の指定

「次へ」をクリックしてテンプレートを選択すると、「プロパティ: コネクタ固有プロパティ・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・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

プロパティの一般特性の選択を終えたら、「値」タブをクリックします。

値の指定

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

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

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

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

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

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

依存関係の設定

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

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

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

!= (等しくない)
> (より大)
< (より小)
>= (より大か等しい)
<= (より小か等しい)

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

新規構成ファイルの作成

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

ブローカーを選択するには、以下のステップを実行します。

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

また、以下の作業も実行できます。

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

コネクタ固有のテンプレートからの構成ファイルの作成

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

1. 「ファイル」>「新規」>「コネクタ構成」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」ダイアログ・ボックス表示されず。

- **名前**

コネクタ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。

- 「コネクタ固有プロパティ・テンプレート」を選択します。

ご使用のコネクタ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・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固有のプロパティが表示されます。

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、コネクタの始動ファイルで指定するコネクタ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. この付録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。

- コネクタ定義ファイル。

コネクタ定義ファイルは、特定のコネクタのプロパティと、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 コネクタの場合は CN_XML.txt です)。

- ICS リポジトリー・ファイル。

コネクタの以前の ICS インプリメンテーションで使用した定義は、そのコネクタの構成で使用されたりポジトリー・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、コネクタのコネクタ固有プロパティのほとんど、あるいはすべてが含まれますが、この付録で後述するように、コネクタ構成ファイルは、ファイルを開いて、プロパティ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を構成するには、Connector Configurator でそのファイルを開き、構成を修正し、そのファイルを再度保管する必要があります。

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

1. Connector Configurator 内で、「ファイル」>「開く」>「ファイルから」をクリックします。
2. 「ファイル・コネクタを開く」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 (*.in、*.out)
ICS 環境でのコネクタの構成にリポジトリ・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・ファイルに複数のコネクタ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構成を開くには、以下のステップを実行します。

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

構成ファイルの完成

構成ファイルを開くか、プロジェクトからコネクタを開くと、「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」タブで、BrokerType プロパティ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が「標準のプロパティ」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 90 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。

複数のコネクタ構成を開いている場合、構成をすべてファイルに保管するには「すべてファイルに保管」を選択し、コネクタ構成をすべて 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すべてに値が設定されているかどうかを確認されます。必須の標準プロパティ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・プロパティの設定

新規のコネクタ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で、以下のカテゴリのプロパティ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
- コネクタ固有のプロパティ
- サポートされているビジネス・オブジェクト
- トレース/ログ・ファイルの値
- データ・ハンドラー (保証付きイベント・デリバリーで JMS メッセージングを使用するコネクタの場合に該当する)

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の場合、以下のプロパティ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値も設定可能です。ただし、標準のプロパティおよびコネクタ固有プロパティ、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。

標準プロパティとコネクタ固有プロパティの違いは、以下のとおりです。

- コネクタの標準プロパティは、コネクタ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有されます。すべてのコネクタが同じ標準プロパティのセットを使用します。これらのプロパティの説明は、各アダプター・ガイドの付録 A にあります。変更できるのはこれらの値の一部のみです。
- アプリケーション固有のプロパティは、コネクタ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。各コネクタには、そのコネクタ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があります。これらのプロパティには、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および推奨値が記述されています。

「標準プロパティ」と「コネクタ固有プロパティ」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を表します。これらのプロパティは、アプリケーションまたはコネクタ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の除去も可能です。
- 「値」フィールドは構成できます。
- 「更新メソッド」フィールドは通知用であり、構成できません。このフィールドは、値が変更されたプロパティをアクティブにするために必要なアクションを示します。

標準コネクタ・プロパティの設定

標準のプロパティの値を変更するには、以下の手順を実行します。

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

を閉じると、すべてのカテゴリ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の設定

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

1. グリッドの左上端の部分で右マウス・ボタンをクリックします。ポップアップ・メニュー・バーが表示されます。プロパティを追加するときは「追加」をクリックします。子プロパティを追加するには、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. プロパティまたは子プロパティの値を入力します。
3. プロパティを暗号化するには、「暗号化」ボックスを選択します。
4. 88 ページの『標準コネクタ・プロパティの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

各プロパティごとに表示される「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・プロパティ名を変更すると、コネクタに障害が発生する可能性があります。コネクタ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、特定のプロパティ名が必要である場合があります。

コネクタ・プロパティの暗号化

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

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

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

更新メソッド

付録『コネクターの標準構成プロパティ』の 62 ページの『プロパティ値の設定と更新』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

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

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

ご使用のブローカーが ICS の場合

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

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

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

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

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

サポートされるリストからビジネス・オブジェクトを削除すると、コネクター定義が変更され、削除されたビジネス・オブジェクトはコネクターのこのインプリメン

ーションで使用不可になります。コネクターのコードに影響したり、そのビジネス・オブジェクト定義そのものが System Manager から削除されることはありません。

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

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

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

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

ほとんどのコネクターの場合、選択可能な項目は「最大限の努力」のみです。

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

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

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

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

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

ご使用のブローカーが WAS の場合

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

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

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

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

関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付けられたマップのリストをサポートします。このリストは、「関連付けられたマップ」タブを選択すると表示されます。

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

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

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

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的**

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

明示的バインディングが必要なのは、特定のサポートされるビジネス・オブジェクトに複数のマップが存在する場合のみです。ICS は、ブート時、各コネクターでサポートされるそれぞれのビジネス・オブジェクトにマップを自動的にバインドしようとしています。複数のマップでその入力データとして同一のビジネス・オブ

ジェクトが使用されている場合、サーバーは、他のマップのスーパーセットである 1 つのマップを見つけて、バインドしようとします。

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

以下の手順を実行して、マップを明示的にバインドします。

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

リソース (ICS)

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

すべてのコネクタがこの機能をサポートしているわけではありません。複数のプロセスを使用するよりも複数のスレッドを使用する方が通常は効率的であるため、Java でマルチスレッドとして設計されたコネクタ・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・プロパティは、DeliveryTransport 標準プロパティの値として 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は、コネクタ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

コネクタ構成ファイルまたはコネクタ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、指定する格納場所へ移動し、ファイル名を指定し、「保管」をクリックします。ログ・メッセージまたはトレース・メッセージは、指定した場所の指定したファイルに書き込まれます。

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

データ・ハンドラー

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

これらのプロパティーに使用する値については、61 ページの『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。
- 指定したディレクトリーに保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに `*.cfg` 拡張子付きファイルとして保管します。

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

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

構成ファイルの変更

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

注: 統合ブローカーを切り替える場合には、ブローカー・モード・プロパティと同様に他の構成プロパティ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」タブを選択します。
- 「標準のプロパティ」タブの「**BrokerType**」フィールドで、ご使用のブローカーに合った値を選択します。

現行値を変更すると、プロパティ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更新され、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 プロパティの値のリストにローカル en_GB を追加するには、stdConnProps.xml ファイルを開き、以下に太文字で示した行を追加してください。

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

付録 C. InFashion サンプルの実行

このシナリオでは、IBM WebSphere Business Integration Adapter for WebSphere Commerce (WebSphere Commerce Adapter) が WebSphere MQ メッセージング・ブローカーを介して WebSphere Commerce Business Edition (WCBE) Application (修正パッケージ 2 が添付されているバージョン 5.4) からビジネス・オブジェクトを送受信するしくみを実例を挙げながら説明します。これは、アダプターの機能性について基本的な点を示した単純なシナリオです。

IBM WebSphere Business Integration Adapter for WebSphere Commerce のインストール

このシナリオには、InFashion ストアがインストール済みかつ稼働中の状態にある WebSphere Commerce アプリケーションが登場します。WebSphere Commerce Application では、定義済みの WebSphere MQ キュー間で注文および顧客に関する情報をやり取りできます。ここでは、ビジネス・オブジェクト

「WCS_Create_WCS_Customer」を作成して、WCBE からの XML メッセージの中に定義されているフィールドを突き合わせてみましょう。このシナリオでは、XML フォーマットのメッセージを WebSphere InterChange Server (ICS) に対して送受信します。

ここでは、WebSphere Commerce Adapter と Port コネクタの間でビジネス・オブジェクトを配信するのに必要なテンプレートおよびコラボレーションを作成します。Port コネクタは、ICS の全インストール先に格納されます。Port コネクタは、基本コード不使用のコネクタ定義だけから構成されているため、コラボレーション用のダミー・コネクタとして機能します。WCBE は起動後に WebSphere MQ メッセージ・ブローカーを使用して WebSphere Commerce Adapter 経由でメッセージを ICS に対して送受信できます。

このシナリオでは、WebSphere Commerce の Infashion ストアで注文を作成してみましょう。これにより、仕入れ注文が WebSphere Commerce からコネクタに送信されます。コネクタは XML DataHandler を使って仕入れ注文をビジネス・オブジェクトに変換してから InterChange サーバーに配信します。Test Connector を使って Port コネクタをシミュレートし、WebSphere Commerce がポストしたビジネス・オブジェクトを取得して Purchase Order 属性を調べましょう。

もう 1 つの別のシナリオでは、Customer ビジネス・オブジェクトを作成して Test Connector から WebSphere Commerce Adapter に送信します。すると、WebSphere Commerce Adapter はそのビジネス・オブジェクトを XML フォーマットに変換して WebSphere Commerce Application に送信します。このことを検証するには、WebSphere Commerce Application 内の顧客レコードを見て調べます。新規に顧客が作成されているはずですが。

前提条件

このシナリオでは、以下のソフトウェアが以下のソフトウェア・インストール前提条件に準拠していることを必要としています。

- ICS および WebSphere Commerce Adapter がインストール済みであること
- WebSphere MQ 5.3 がインストール済みかつ稼働中の状態にあること

このシナリオでは、以下の WebSphere Commerce Adapter キューを以下の手順で構成しておくことも必要です。

- コマンド・プロンプトから、以下のディレクトリーに切り替えます。
%CrossWorlds%\connectors\WebSphereCommerce\utilities
- ConfigureAdapter.bat <InterChangeServerName>. queue.manager を実行します。
- 「プログラム <ICS 名> Queue Manager の構成 (Programs <ICS Name> Configure Queue Manager)」で、IBM WebSphere InterChange Server の Configure Queue Manager コマンドを実行します。
- IBM WebSphere Commerce JMS キューを WebSphere MQ で動作するように構成します。
- これで、WebSphere Commerce JMS キューが WebSphere MQ で動作するように構成されました。

サンプル・シナリオのインストール

以下のセクションでは、WebSphere Commerce Server とシミュレート済みバックエンド間の往復を可能にするシナリオのインストール、構成、およびテストを行います。

注: %SAMPLE_FOLDER% への参照は%connectors%\WebSphereCommerce\Samples にマップされます。%CROSSWORLDS% は IBM WebSphere InterChange Server のインストール内容が入っているフォルダーにマップされます。

サンプル・シナリオをインストールするには、以下のステップを使用します。

1. 永続 IOR ファイルを作成します。
 - Interchange Server が現在稼働中の状態にある場合は停止させます。
 - WebSphere InterChange Server ディレクトリーにある <Interchange Server 構成> ファイルを編集します。この構成ファイルの名称は、通常「InterchangeSystem.cfg」です。以下の行を追加します。
CORBA[]
OAPORT=16000
 - WebSphere InterChange Server を始動します。
2. ポート・コネクタを检查します。
 - WebSphere InterChange Server の System Manager を使用して、ポート・コネクタの定義がロードされたことを確認します。ポート・コネクタは通常、標準の WebSphere InterChange Server インストール内容と一緒に提供されますが、手動で作成することもできます。Service Manager で別のコネクタを作成して「PortConnector」という名前を付けるだけです。

- repos_copy を使用して、ファイルに含まれているビジネス・オブジェクトをロードします。

```
%SAMPLE_FOLDER%/WC_PortConnector.in
```

- コマンド・プロンプトで、例えば以下のように入力します。

```
repos_copy -iWC_PortConnector.in -s <ICS_Server_Name>
```

3. ビジネス・オブジェクトをリポジトリにロードします。

repos_copy を使用して、ファイルに含まれているビジネス・オブジェクトをロードします。

```
%SAMPLE_FOLDER%/WC_BODefinition.in
```

- コマンド・プロンプトで、例えば以下のように入力します。

```
repos_copy -s<InterChangeServerName> -u<UserName>  
-p<password> -iWC_BODefinition.in
```

repo が正常にロードされた場合、「リポジトリがロードされました (Repository Loaded)」という確認メッセージが表示されます。

注: Edifecs SpecBuilder または XML Borgen ユーティリティを使用して、独自の BO 定義を作成することもできます。BO 定義の作成を行うには、IBM WebSphere Commerce アプリケーションから IBM WebSphere InterChange Server までの送受信が可能な XML 文書の DTD を使用します。

4. テンプレート定義をリポジトリにロードします。

repos_copy を使用して、ファイルに含まれているテンプレートをロードします。

```
%SAMPLE_FOLDER%/WC_CollabTemplate.in
```

- コマンド・プロンプトで、例えば以下のように入力します。

```
repos_copy -s<InterChangeServerName> -u<UserName>  
-p<password> -iWC_CollabTemplate.in
```

5. テンプレートのクラス・ファイルを追加します。

コピーするファイル:

```
%SAMPLE_FOLDER%/CollaborationClasses/*.class
```

コピー先フォルダー:

```
%CROSSWORLDS%/collaborations/classes/UserCollaborations
```

6. コラボレーションをリポジトリにロードします。

repos_copy を使用して、ファイル %SAMPLE_FOLDER%/WC_CollabObj.in に含まれているコラボレーションをロードします。

注: コラボレーションをロードする前に、テンプレート・ファイルをロードしておく必要があります。

コマンド・プロンプトで、例えば以下のように入力します。

```
repos_copy -s<InterChangeServerName> -u<UserName>
-p<password> -iWC_CollabObj.in
```

7. IBM WebSphere Commerce 構成の静的メタオブジェクトをロードします。

repos_copy を使用して、ファイル

%SAMPLE_FOLDER%/MO_WebSphereCommerceConfig.in に含まれているコラボレーションをロードします。

コマンド・プロンプトで、例えば以下のように入力します。

```
repos_copy -s<InterChangeServerName> -u<UserName>
-p<password> -iMO_WebSphereCommerceConfig.in
```

8. コネクタを構成します。

System Manager を開き、「WebSphere Commerce」コネクタをダブルクリックして WebSphereCommerceConnector を調べます。コネクタがまだ構成されていない場合は、インストール・ガイドに記載されている手順に従って構成します。さらに、以下のコネクタ・プロパティが以下の値と一致するかどうかの検査や変更も行います。

表 19. 構成プロパティ

プロパティ	値
ConfigurationMetaObject	MO_WebSphereCommerceConfig
DataHandlerConfigMO	MO_DataHandlerDefault
DataHandlerMimeType	text/xml
ErrorQueue	queue://<queue manager name>/WC_MQCONN.ERROR
InProgressQueue	queue://<queue manager name>/WC_MQCONN.IN_PROGRESS
ReplyToQueue	queue://<queue manager name>/WC_MQCONN.IN_REPLY
ArchiveQueue	queue://<queue manager name>/WC_MQCONN.ARCHIVE
InputQueue	queue://<queue manager name>/WCS_OUTBOUND
UnsubscribedQueue	queue://<queue manager name>/WC_MQCONN.UNSUBSCRIBE
Hostname	WebSphere MQ が実行されているコンピューターのホスト名。
Channel	Queue Manager 用に構成されたチャネル。デフォルト値は CHANNEL1 です。

9. ビジネス・オブジェクトに対するサポートを有効にします。

ビジネス・オブジェクトを使用するためには、まずコネクタがそのビジネス・オブジェクトをサポートしていなければなりません。

- System Manager を使用して、WebSphere Commerce Adapter の定義を開きます。
- 「サポートされているビジネス・オブジェクト」タブを選択して、以下のビジネス・オブジェクトおよび PortConnector を追加します。また、PortConnector 用にサポートされているビジネス・オブジェクトがすべて WebSphere InterChange Server に存在していることを確認します。
 - Name=MO_WebSphereCommerceConfig
 - Name=MO_DataHandlerDefault
 - Name=MO_DataHandlerDefault
 - Name=WCS_Create_WCS_Customer
 - Name=WCS_Report_NC_PurchaseOrder

各オブジェクトおよび PortConnector 用の「AgentSupport」チェック・ボックスをクリックします。

10. MO_WebSphereCommerceConfig 内のアプリケーション固有のビジネス・オブジェクトのサポートを確認します。

MO_WebSphereCommerceConfig 内のビジネス・オブジェクトを表しているアプリケーション固有の属性は、適切な出力キューを指していなければなりません。このシナリオでは、属性「WCS_Create_WCS_Customer」に以下のようなアプリケーション固有の値を設定しておく必要があります。

```
queue://<queue manager name>/WCS_Serial_Inbound?targetClient=1;OutputFormat=MQSTR
```

11. WebSphere InterChange Server を再始動します。

WebSphere InterChange Server をリブートして、変更内容をすべてインスタンス化します。

12. コラボレーションおよびコネクタを始動します。

コラボレーションおよびコネクタがただちに始動しない場合は、メニューを右クリックして「始動 (Start)」を選択します。

シナリオの実行

シナリオを実行するには、以下のステップを使用します。

1. WebSphere Components を始動します。

Start the WebSphere InterChange Server および WebSphere Commerce エージェントがまだ稼働していない場合は始動します。

2. ポート・コネクタをシミュレートします。

Test Connector を使用して、「ファイル」メニューから「エージェントの接続」を選択し、「PortConnector」のプロファイルを定義します。これにより、WebSphere Commerce Server と対話するバックエンド・アプリケーションがシミュレートされます。

3. WebSphere Commerce Server アプリケーションを始動します。

- 「WebSphere Application Server」を強調表示し、「プログラム」メニューから「始動 (Start)」を選択して、WebSphere Application Server を始動します。
- 「WebSphere Application Server 管理コンソール (WebSphere Application Server Admin Console)」を強調表示し、「プログラム」メニューから「始動 (Start)」を選択して、管理コンソールを始動します。

4. 管理コンソール内から、サーバー名およびアプリケーション・サーバーのノードを展開します。

- 管理コンソール内から、サーバー名およびアプリケーション・サーバーのノードを展開し、以下のサーバーを始動します。

– Payment Manager Server (以下のように入力します。)

```
<Payment Server Installation Name>¥IBMPayServer
```

– WebSphere Commerce Server

5. PortConnector から WCS までのフローをテストします。

TestConnect を使用して接続をテストします。

- Test Connector を使用して「WCS_Create_WCS_Customer」ビジネス・オブジェクトの新規インスタンスを作成します。動詞を「Create」に設定し、メッセージを送信します。
6. WebSphere Commerce Server にメッセージが配信されたことを検証します。
- PortConnector から送信された顧客情報が WebSphere Commerce Server に正常に配信されたことを検証します。
- 「スタート」>「プログラム」>「IBM WebSphereCommerce」>「WebSphere Commerce Accelerator」から「操作 (Operations)」を選択します。
 - 「操作 (Operations)」から「顧客の検索 (Find Customer)」を選択します。
 - WebSphere Commerce Server ストアにログインして、ユーザーが作成されたことを検証します。
 - Mall データベースに SQL の照会を使用して、Users テーブル内のユーザーを検証します。
7. WebSphere Commerce Server から PortConnector までのフローをテストします。
- 以下の情報を使用して、Commerce Server から PortConnector までのフローをテストします。
- 以下の場所からストアにアクセスします。
`/webapp/wcs/stores/store_directory/index.jsp`
 - WebSphere Commerce Server を使って、プロンプトに従って注文を作成します。すると、バックエンド・システムに送信する必要がある注文が起動されません。
8. PortConnector にメッセージが配信されたことを検証します。
- PortConnector で、メッセージを表示して注文データを検証できます。

アダプターのトラブルシューティング

以下のトピックを使用して、アダプターのトラブルシューティングを行います。

IBM WebSphere Business Integration Adapter が始動しない場合

- アダプター・プロパティ内の「ホスト名」プロパティが正しく構成されていることを検証します。
- WebSphere MQ コンソールを介して、WebSphere MQ リスナーが作成、始動されたことを検証します。
- アダプター構成内のキュー・マネージャー名がすべて正しいことを検証します。
- 構成メタオブジェクトがロードされたことを検証します。このメタオブジェクトは、WebSphere MQ に作成されたキュー・マネージャーの中のメタオブジェクトと同じものでなければなりません。
- パス内に mqjbnd02.d11 が含まれていることを検証します。

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を提供し、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せんが、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をお願いします。

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ます、単に目標を示しているものです。

プログラミング・インターフェース情報

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

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

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

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

商標

以下は、IBM Corporation の商標です。

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

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

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

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。



IBM WebSphere Business Integration Adapter Framework V2.4



Printed in Japan