

**IBM WebSphere Business Integration Server
Express and Express Plus**



Adapter for WebSphere MQ ユーザーズ・ガイド

アダプター・バージョン 2.7.x

**IBM WebSphere Business Integration Server
Express and Express Plus**



Adapter for WebSphere MQ ユーザーズ・ガイド

アダプター・バージョン 2.7.x

お願い

本書および本書で紹介する製品をご使用になる前に、111 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere Business Integration Adapter for WebSphere MQ (5724-H06) バージョン 2.7.x に適用されません。

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

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

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

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

(URL は、変更になる場合があります)

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

原 典： WebSphere Business Integration Server Express and Express Plus
Adapter for WebSphere MQ User Guide
Adapter Version 2.7.x

発 行： 日本アイ・ピー・エム株式会社

担 当： ナショナル・ランゲージ・サポート

第1刷 2005.8

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

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

© Copyright International Business Machines Corporation 2004, 2005. All rights reserved.

© Copyright IBM Japan 2005

目次

本書について	v
対象読者	v
本書の前提条件	v
関連文書	v
表記上の規則	vi
本リリースの新機能	vii
リリース 2.7.x での新機能	vii
第 1 章 概要	1
Adapter for WebSphere MQ の環境	2
コネクタ・アーキテクチャ	3
アプリケーションとコネクタ間の通信方法	4
イベント処理	6
保証付きイベント・デリバリー	10
ビジネス・オブジェクト要求	11
動詞の処理	11
共通の構成タスク	16
第 2 章 アダプターのインストールおよび構成	21
インストール作業の概要	21
アダプターおよび関連ファイルのインストール	21
インストール済みファイルの構造	21
コネクタ構成	23
複数コネクタ・インスタンスの作成	30
キューの Uniform Resource Identifier (URI)	32
メタオブジェクトの構成	33
始動ファイルの構成	46
始動	46
コネクタの停止	47
第 3 章 ビジネス・オブジェクトの作成および変更	49
アダプターのビジネス・オブジェクトの構造	49
エラー処理	52
トレース	53
第 4 章 トラブルシューティング	55
始動時の問題	55
イベント処理	55
付録 A. コネクタの標準構成プロパティ	57
新規プロパティ	57
標準コネクタ・プロパティの概要	57
標準プロパティの早見表	59
標準プロパティ	65
付録 B. Connector Configurator	83
Connector Configurator Express の概要	83
Connector Configurator Express の始動	84
System Manager からの Configurator の実行	85

コネクター固有のプロパティ・テンプレートの作成	85
新しい構成ファイルを作成	88
既存ファイルの使用	90
構成ファイルの完成	91
構成ファイル・プロパティの設定	92
構成ファイルの保管	100
構成ファイルの変更	100
構成の完了	101
グローバル化環境における Connector Configurator Express の使用	101
付録 C. チュートリアル	103
チュートリアルについて	103
始める前に	104
環境のセットアップ	104
シナリオの実行	107
特記事項	111
プログラミング・インターフェース情報	112
商標	113
索引	115

本書について

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

この資料では、IBM WebSphere Business Integration Server Express Adapter for WebSphere MQ のインストール、コネクタ・プロパティ構成、ビジネス・オブジェクトの開発、およびトラブルシューティングについて説明します。

特に明記されていない限り、本書の情報は、いずれも、IBM WebSphere Business Integration Server Express と IBM WebSphere Business Integration Server Express Plus の両方に当てはまります。「WebSphere Business Integration Server Express」という用語と、これを言い換えた用語は、これらの 2 つの製品の両方を指します。

対象読者

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

本書の前提条件

本書の読者は、WebSphere Business Integration システム、ビジネス・オブジェクトとコラボレーションの開発、および WebSphere MQ アプリケーション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。リンクについては、『関連文書』を参照してください。

関連文書

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

文書は、以下のサイトでダウンロード、インストール、および表示できます。
<http://www.ibm.com/websphere/wbiserverexpress/infocenter>

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、ユーザーの入力した情報、システムが画面に出力した情報など、記述されたとおりの値を示します。
太字	初出語を示します。
イタリック、イタリック青のアウトライン	変数名または相互参照を示します。 青のアウトラインは、マニュアルをオンラインで表示するときのみ見られるもので、相互参照用のハイパーリンクを示します。アウトラインの内側をクリックすることにより、参照先オブジェクトにジャンプできます。
{ }	構文の記述行の場合、中括弧 { } で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
[]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプション・パラメーターです。
...	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、option[,...] は、複数のオプションをコンマで区切って入力できることを示します。
< >	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。例えば、<server_name><connector_name>tmp.log のように使用します。
/, ¥	本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。Linux ^(R) および i5/OS インストール済み環境の場合には、円記号をスラッシュ (/) 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。
%text% と \$text	% 記号で囲まれたテキストは、Windows ^(R) の text システム変数またはユーザー変数の値を示します。Linux 環境においてこれに相当する表記は \$text で、text Linux 環境変数の値を示します。
ProductDir	製品のインストール先ディレクトリーを表します。各プラットフォームのデフォルトは、以下のとおりです。 Windows: IBM¥WebSphereServer Linux: /home/\${username}/IBM/WebSphereServer i5/OS: /QIBM/ProdData/WBIServer44/product

本リリースの新機能

リリース 2.7.x での新機能

アダプター・バージョン 2.7.x について述べた本書のこのリリースには、以下の新しいまたは訂正された情報が含まれています。

このリリースでは、双方向スクリプト・データの処理をサポートしています。詳細については、2 ページの『ロケール依存データ』を参照してください。

このリリースでは、以下のプラットフォームまたは更新のサポートを追加しています。

- Microsoft Windows 2003 (Standard Edition または Enterprise Edition)
- Linux:
 - RedHat Enterprise Linux WS/AS/ES 3.0 Update 2, Intel (IA32)
 - SuSE Linux ES 8.1 SP3, Intel (IA32) SP
 - SuSE Linux ES 9.0, Intel (IA32)
- i5/OS V5R3 および OS/400 V5R2
- カスタム・アダプターをコンパイルするために Java コンパイラー (Windows 2000 用 IBM JDK 1.4.2) を必要とするすべてのオペレーティング・システム環境

コネクター固有プロパティ `ReplyToQueuePollFrequency` が文書化されました。

このリリースでは、トレース・レベル 5 を使用し、アダプターによってキャッチされた例外の `printStackTrace()` をダンプする方法をサポートしています。

`OutputFormat` メタオブジェクト・プロパティの名前の長さの制限が 16 文字であることが文書化されました。

第 1 章 概要

- 2 ページの『Adapter for WebSphere MQ の環境』
- 3 ページの『コネクタ・アーキテクチャ』
- 4 ページの『アプリケーションとコネクタ間の通信方法』
- 6 ページの『イベント処理』
- 10 ページの『保証付きイベント・デリバリー』
- 11 ページの『ビジネス・オブジェクト要求』
- 11 ページの『動詞の処理』
- 16 ページの『共通の構成タスク』

WebSphere MQ 用のコネクタは、WebSphere Business Integration Server Express Adapter for WebSphere MQ のランタイム・コンポーネントの 1 つです。このコネクタを使用すると、WebSphere 統合ブローカーと、WebSphere MQ メッセージの形式でデータを送受信するアプリケーションとの間で、ビジネス・オブジェクトを交換できます。この章では、コネクタ・コンポーネントおよび関連するビジネス・インテグレーション・システム・アーキテクチャについて説明します。

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

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

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

統合ブローカーとコネクタの関係に関する詳細については、「システム管理ガイド」を参照してください。

注: すべての WebSphere Business Integration Server Express アダプターは、統合ブローカーと連携して動作します。WebSphere MQ 用のコネクタは、InterChange Server Express 統合ブローカーと連動します。統合ブローカーについては、「システム・インプリメンテーション・ガイド」を参照してください。

Adapter for WebSphere MQ の環境

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

- 『アダプター・プラットフォーム』
- 『アダプターの依存関係』
- 『ロケール依存データ』

アダプター・プラットフォーム

ブローカーのほかに、このアダプターでは以下のオペレーティング・システムのいずれかが必要です。

- Microsoft Windows 2003 (Standard Edition または Enterprise Edition)
- Linux:
 - RedHat Enterprise Linux WS/AS/ES 3.0 Update 2、Intel (IA32)
 - SuSE Linux ES 8.1 SP3、Intel (IA32)
 - SuSE Linux ES 9.0、Intel (IA32)
- IBM i5/OS V5R3 および OS/400 V5R2

注: 明示されない限り、I5/OS は i5/OS および OS/400 を意味します。

アダプターの依存関係

このアダプターには、以下のソフトウェア前提条件と、その他の依存関係があります。

- コネクタがアプリケーションとのインターオペラビリティをサポートするために使用できるソフトウェアは WebSphere MQ 5.1、5.2¹、および 5.3 を介したアプリケーションとのインターオペラビリティをサポートします。そのため、これらのいずれかのソフトウェア・バージョンをインストールしている必要があります。
- さらに、IBM WebSphere MQ Java クライアント・ライブラリーも必要です。

注: このアダプターは、WebSphere MQ 5.3 環境で SSL (Secure Socket Layer) をサポートしていません。アダプター・フレームワークと統合ブローカーの通信に適切な WebSphere MQ ソフトウェア・バージョンについては、Windows、Linux、または i5/OS 用の「WebSphere Business Integration Server Express インストール・ガイド」を参照してください。

ロケール依存データ

コネクタは国際化され、2 バイト文字セットをサポートし、特定の言語でメッセージ・テキストを配信できるようになっています。コネクタが、ある文字コード・セットを使用する場所から別の文字コード・セットを使用する場所にデータを転送する場合、文字変換を実行してデータの意味を保持します。

1. ご使用の環境で文字セット変換に convert-on-the-get 方法を実装している場合は、最新の MA88 (JMS クラス) を IBM からダウンロードしてください。パッチ・レベルは最低でも 5.2.2 である必要があります (WebSphere MQ バージョン 5.2 の場合)。これにより、サポートされないエンコード・エラーを避けることができます。

このアダプターは、アラビア語、ヘブライ語、ウルドゥー語、ペルシア語、およびイディッシュ語などの言語に対して、双方向スクリプト・データの処理をサポートします。双方向の能力を使用するには、標準の双方向プロパティを構成する必要があります。詳しくは、付録 A の『コネクターの標準構成プロパティ』を参照してください。

Java 仮想マシン (JVM) 内での Java ランタイム環境は、Unicode 文字コード・セットでデータを表します。Unicode には、ほとんどの既知の文字コード・セット (1 バイト系とマルチバイト系をいずれも含む) に対応できるエンコード方式が組み込まれています。WebSphere Business Integration システムの大部分のコンポーネントは Java で作成されています。したがって、ほとんどの統合コンポーネントの間でデータが転送されても、文字変換の必要はありません。

エラー・メッセージや通知メッセージを個々の国や地域に合った適切な言語で記録するには、個々の環境に合わせて Locale 標準構成プロパティを構成する必要があります。構成プロパティの詳細については、57 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。

コネクター・アーキテクチャー

コネクターはメタデータ主導型です。メッセージのルーティングおよびフォーマット変換は、イベント・ポーリング技法によって開始されます。コネクターは、エンタープライズ・メッセージング・システムへのアクセスの API である Java™ Message Service (JMS) の IBM WebSphere MQ インプリメンテーションを使用します。エンタープライズ・メッセージング・システムは、保証されたイベント配信を可能にします。

コネクターを使用すると、IBM WebSphere Business Integration Collaborations と、データの変更が発生したときに WebSphere MQ メッセージを発行または受信するアプリケーションとの間で、非同期的にビジネス・オブジェクトを交換できます。

コネクターはキューから WebSphere MQ メッセージを検索し、データ・ハンドラーを呼び出してメッセージを対応するビジネス・オブジェクトに変換し、コラボレーションにデリバリーします。反対方向の場合、コネクター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、同じデータ・ハンドラーを使用して WebSphere MQ メッセージに変換し、WebSphere MQ キューにデリバリーします。

コネクターは、任意のデータ・ハンドラーを使用してメッセージを処理するように構成できます。詳細については、「データ・ハンドラー・ガイド」を参照してください。

メッセージの処理に使用されるビジネス・オブジェクトのタイプと動詞は、WebSphere MQ メッセージ・ヘッダーに含まれる FORMAT フィールドによって決定されます。コネクターは、メタオブジェクト・エンTRIESを使用してオブジェクト名と動詞を決定します。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構成し、WebSphere MQ メッセージ・ヘッダーの FORMAT フィールドのテキストに関連付けます。

オプションで動的メタオブジェクトを構成し、コネクターに渡されるビジネス・オブジェクトの子として追加することもできます。この子メタオブジェクトの値は、

コネクタ全体に対して指定されている静的メタオブジェクトの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場合、または子メタオブジェクトが必要な変換プロパティを定義していない場合、デフォルトでは、コネクタは静的メタオブジェクトの値を調べます。1 つの静的コネクタ・メタオブジェクトの代わりに、またはその補足として、1 つ以上の動的子メタオブジェクトを指定できます。

コネクタは複数の入力キューをポーリングできます。その際、各入力キューをラウンドロビン方式でポーリングし、各入力キューから指定された数のメッセージを検索します。コネクタは、ポーリング中に検索された各メッセージに、動的子メタオブジェクト (ビジネス・オブジェクトで指定されている場合) を追加します。子メタオブジェクトの値は、コネクタに対し、メッセージのフォーマットおよびメッセージが検索された入力キューの名前を属性に取り込むように指示できます。

入力キューからメッセージが検索されると、コネクタは、その入力キューと、メッセージ・ヘッダーに含まれる `FORMAT` フィールドに関連付けられているビジネス・オブジェクト名を調べます。次に、そのビジネス・オブジェクトの新しいインスタンスと共に、メッセージの本体がデータ・ハンドラーに渡されます。入力キューおよびフォーマットに関連付けられているビジネス・オブジェクト名がない場合は、メッセージの本体だけがデータ・ハンドラーに渡されます。ビジネス・オブジェクトにメッセージの内容が正常に取り込まれると、コネクタはそのビジネス・オブジェクトがサブスクライブされているかどうかをチェックしてから、`getAppEvents()` メソッドを使用して `InterChange Server Express` にデリバリーします。

アプリケーションとコネクタ間の通信方法

コネクタは、`IBM WebSphere MQ` に実装されている `Java Message Service (JMS)` を使用して通信します。`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 API です。`JMS` 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的に送受信できるように設計されています。

メッセージ要求

図 1 に、メッセージ要求の通信を示します。`doVerbFor()` メソッドがコラボレーションから `WebSphere Business Integration` システムのビジネス・オブジェクトを受け取ると、コネクタはそのビジネス・オブジェクトをデータ・ハンドラーに渡します。データ・ハンドラーはそのビジネス・オブジェクトを `JMS` に適したテキストに変換し、コネクタがそれをメッセージとしてキューに送ります。このとき、`JMS` 層は適切な呼び出しを実行してキュー・セッションを開き、メッセージの経路を指定します。

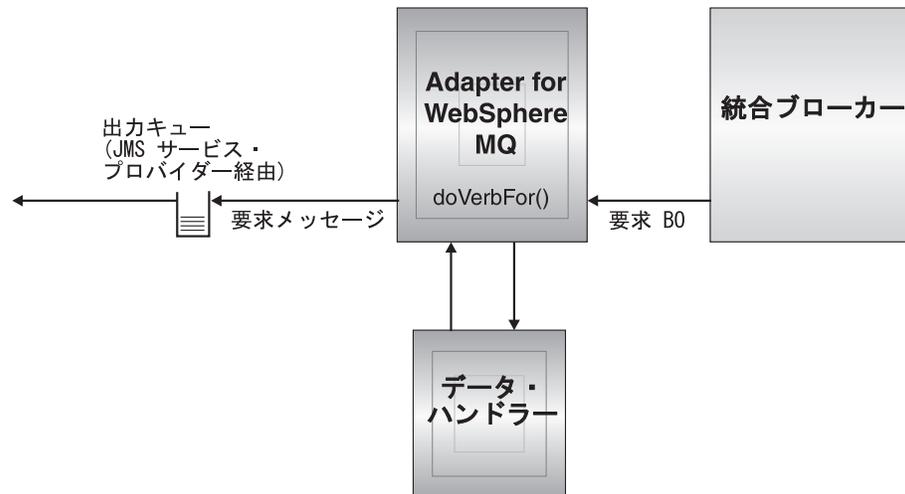


図1. アプリケーションとコネクタの間の通信方法: メッセージ要求

イベント・デリバリー

図2 に、イベント・デリバリーの方向を示します。 `pollForEvents()` メソッドは、次の該当するメッセージを入力キューから検索します。メッセージは実行中のキューに入れられ、処理が完了するまでキュー内に残ります。コネクタは最初に、静的メタオブジェクトまたは動的メタオブジェクトのいずれかを使用して、そのメッセージ・タイプがサポートされているかどうかを調べます。サポートされている場合、コネクタは構成されているデータ・ハンドラーにメッセージを渡し、データ・ハンドラーがそれを WebSphere Business Integration システムのビジネス・オブジェクトに変換します。設定される動詞には、そのメッセージ・タイプに対して定義されている変換プロパティが反映されます。次に、コネクタは、そのビジネス・オブジェクトがコラボレーションによってサブスクライブされているかどうかを調べます。サブスクライブされている場合、`getAppEvents()` メソッドがビジネス・オブジェクトを InterChange Server Express にデリバリーし、実行中のキューからメッセージが削除されます。

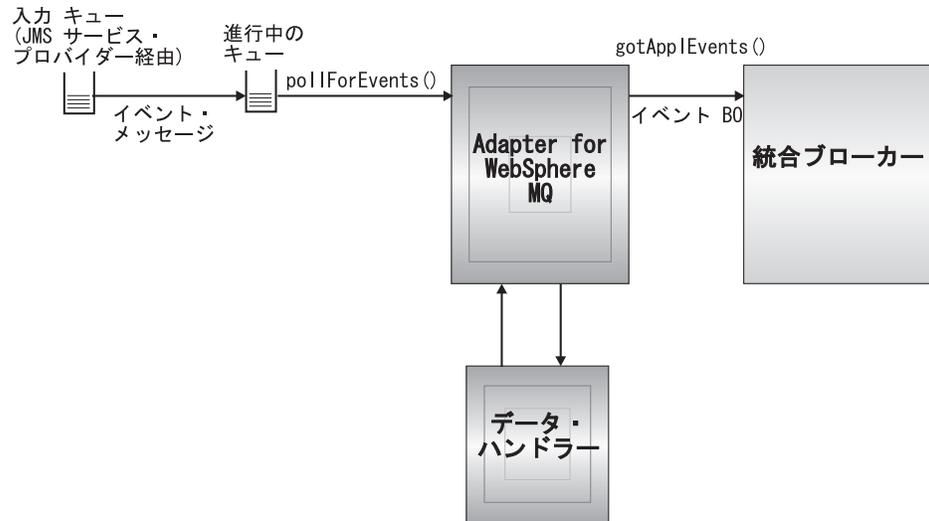


図2. アプリケーションとコネクタの間の通信方法: イベント・デリバリー

イベント処理

コネクタは、イベント通知のために、データベース・トリガーではなくアプリケーションによってキューに書き込まれたイベントを検出します。イベントは、アプリケーションまたはその他の MQ 対応ソフトウェアが WebSphere MQ メッセージを生成して MQ メッセージ・キューに格納するときが発生します。

検索

コネクタは、`pollForEvents()` メソッドを使用して MQ キューからメッセージを定期的にポーリングします。メッセージを検出すると、コネクタはそれを MQ キューから検索して調べ、メッセージのフォーマットを判別します。判別されたフォーマットがコネクタの静的オブジェクトで定義されている場合、コネクタは、メッセージの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・ハンドラーはビジネス・オブジェクトを取り込み、動詞を指定すると想定されています。判別された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、コネクタはメッセージの本体のみをデータ・ハンドラーに渡します。データ・ハンドラーはメッセージに対する正しいビジネス・オブジェクトを判別して作成し、取り込むと想定されています。イベント失敗のシナリオについては、52 ページの『エラー処理』を参照してください。

コネクタは、最初に入力キューとのトランザクション・セッションを開いて、メッセージを処理します。このトランザクション・アプローチを使用すると、コネクタがビジネス・オブジェクトを正常にサブミットしたにもかかわらず、キューでトランザクションをコミットすることに失敗した場合に、コラボレーションにビジネス・オブジェクトが 2 回デリバリーされてしまう可能性が若干あります。この問題を回避するために、コネクタはすべてのメッセージを実行中のキューに移動します。その結果、メッセージは、処理が完了するまでキュー内に保留されます。処

理中にコネクタが予期しないエラーでシャットダウンした場合、メッセージは元の入力キューには復元されず、実行中のキュー内に残されます。

注: JMS サービス・プロバイダーとのトランザクション・セッションでは、キュー上の要求されたすべての処理が、キューからイベントが削除される前に実行され、コミットされる必要があります。したがって、コネクタがキューからメッセージを検索するときには、次の 3 つの処理が実行されるまでは検索がコミットされません。1) メッセージからビジネス・オブジェクトへの変換、2) `getApplEvents()` メソッドによる、InterChange Server Express へのビジネス・オブジェクトのデリバリー、および 3) 戻り値の受信。

同期イベント処理

WebSphere MQ 対応コネクタは、WebSphere MQ を使用して発行した要求に関するフィードバックを必要とするアプリケーションをサポートするために、オプションでレポート・メッセージをアプリケーションに返送します。このレポート・メッセージには、アプリケーションからの要求の処理が完了した時点での結果が詳細に記述されます。

この処理を実現するために、コネクタはこのような要求のビジネス・データを InterChange Server Express に同期的に通知します。コラボレーションがビジネス・オブジェクトを正常に処理した場合、コネクタは、InterChange Server Express からの戻りコードとビジネス・オブジェクトのすべての変更を含むレポートを要求発行者に返送します。コネクタまたはコラボレーションがビジネス・オブジェクトの処理に失敗した場合、コネクタは、該当するエラー・コードとエラー・メッセージを含むレポートを返送します。

いずれの場合も、WebSphere MQ 対応コネクタに要求を送信するアプリケーションは、要求の結果について通知されます。

処理: WebSphere MQ 対応コネクタが肯定通知レポートまたは否定通知レポート (PAN または NAN) を要求するメッセージを受け取った場合、コネクタはそのメッセージの内容を InterChange Server Express に同期的に通知し、レポート・メッセージに戻りコードと変更されたビジネス・データを組み込んで、要求を発行したアプリケーションに返送します。

次の表に、コネクタに送信された WebSphere MQ メッセージが同期的に処理されるために必要な構造を示します。

MQMD フィールド	説明	サポートされる値 (複数の値をサポートするには論理和を使用)
MessageType	メッセージ・タイプ	DATAGRAM

MQMD フィールド	説明	サポートされる値 (複数の値をサポートするには論理和を使用)
report	要求されたレポート・メッセージのオプション	<p>次のいずれか一方、または両方を指定できません。</p> <ul style="list-style-type: none"> MQRO_PAN コネクターは、ビジネス・オブジェクトが正常に処理された場合にレポート・メッセージを送信します。 MQRO_NAN コネクターは、ビジネス・オブジェクトの処理中にエラーが発生した場合にレポート・メッセージを送信します。 <p>次のいずれかの値を指定すると、レポート・メッセージの相関 ID の設定方法を制御できます。</p> <ul style="list-style-type: none"> MQRO_COPY_MSG_ID_TO_CORREL_ID コネクターは、要求メッセージのメッセージ ID をレポートの相関 ID にコピーします。これはデフォルトのアクションです。 MQRO_PASS_CORREL_ID コネクターは、要求メッセージの相関 ID をレポートの相関 ID にコピーします。
ReplyToQueue	応答キューの名前	レポート・メッセージの送信先となるキューの名前。
replyToQueueManager	キュー・マネージャーの名前	レポート・メッセージの送信先となるキュー・マネージャーの名前。
メッセージの本体		コネクターに構成されているデータ・ハンドラーと互換性のあるフォーマットで直列化されたビジネス・オブジェクト。

上記の表で説明したメッセージを受け取ると、コネクターは以下の処理を行います。

1. 構成されているデータ・ハンドラーを使用して、メッセージの本体に含まれるビジネス・オブジェクトを再構成します。
2. 静的メタデータ・オブジェクト (コラボレーション名を設定できない動的子メタオブジェクトは除く) のビジネス・オブジェクトおよび動詞のために指定されたコラボレーション名を検索します。
3. 指定されたコラボレーションに、ビジネス・オブジェクトを同期的に通知します。
4. 処理の結果とビジネス・オブジェクトのすべての変更またはエラー・メッセージをカプセル化したレポートを生成します。
5. 要求の replyToQueue および replyToQueueManager フィールド内で指定されたキューに、レポートを送信します。

次の表に、コネクターから要求発行者に返送されるレポートの構造を示します。

MQMD フィールド	説明	サポートされる値 (複数の値がある場合は OR として扱います)
MessageType	メッセージ・タイプ	REPORT

MQMD フィールド	説明	サポートされる値 (複数の値がある場合は OR として扱います)
feedback	レポートのタイプ	次のいずれかです。 <ul style="list-style-type: none"> • MQRO_PAN コラボレーションがビジネス・オブジェクトを正常に処理した場合に、レポートが返送されます。 • MQRO_NAN 要求の処理中にコネクタまたはコラボレーションがエラーを検出した場合に、レポートが返送されます。コラボレーションがビジネス・オブジェクトを正常に処理した場合、コネクタはメッセージの本体にコラボレーションから戻されたビジネス・オブジェクトを取り込みます。このデフォルトの動作は、静的メタデータ・オブジェクトの DoNotReportBusObj プロパティを true に設定することによりオーバーライドできます。 <p>要求を処理できなかった場合、コネクタはメッセージの本体にコネクタまたはコラボレーションによって生成されたエラー・メッセージを取り込みます。</p>
メッセージの本体		

リカバリー

コネクタは初期化の際に実行中のキューを調べ、コネクタのシャットダウンが原因で未処理のまま残っているメッセージがないかどうかを調べます。コネクタの構成プロパティ `InDoubtEvents` を使用すると、そのようなメッセージのリカバリー処理に関する 4 つのオプション (`fail on startup`、`reprocess`、`ignore`、または `log error`) のうち、いずれかを指定できます。

始動時の失敗 (Fail on startup)

`fail on startup` オプションを指定した場合、コネクタが初期化の際、実行中のキュー内にメッセージを検出すると、コネクタはエラーを記録し、即時にシャットダウンします。ユーザーまたはシステム管理者は、検出されたメッセージを調べ、これらのメッセージを完全に削除するかまたは別のキューに移動するなどの適切な処置を取る必要があります。

再処理 (Reprocess)

`reprocessing` オプションを指定した場合、コネクタが初期化の際、実行中のキュー内にメッセージを検出すると、コネクタは以降のポーリングでそのメッセージを最初に処理します。実行中のキュー内にあったすべてのメッセージの処理が完了すると、コネクタは入力キューからのメッセージの処理を開始します。

無視 (Ignore)

`ignore` オプションを指定した場合、初期化の際、コネクタが実行中のキュー内にメッセージを検出すると、コネクタはそれを無視しますが、シャットダウンはしません。

エラー・ログ記録 (Log error)

log error オプションを指定した場合、初期化の際、コネクターが実行中のキュー内にメッセージを検出すると、コネクターはエラーを記録しますが、シャットダウンはしません。

アーカイブ

コネクターのプロパティ `ArchiveQueue` が指定されており、かつ有効なキューを示している場合には、コネク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・キューに格納します。 `ArchiveQueue` が未定義の場合、メッセージは処理後に破棄されます。アンサブスクライブされたメッセージまたはエラーを含むメッセージのアーカイブの詳細については、52 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、検索したメッセージを即時に別のキューに送信することはできません。メッセージをアーカイブして再デリバリーできるようにするために、コネクターは、オリジナルのメッセージから本体とヘッダー (該当する場合のみ) を複製した第 2 のメッセージを最初に生成します。JMS サービス・プロバイダーとの競合を避けるため、JMS に必須のフィールドのみが複製されます。したがって、`format` フィールドは、アーカイブまたは再デリバリーされるメッセージにコピーされる唯一の追加メッセージ・プロパティとなります。

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、コネクター・フレームワークは、コネクターのイベント・ストア、JMS イベント・ストア、および宛先の JMS キューとの間で、イベントを失ったり 2 度送信したりせずに、確実に送信することができます。JMS 対応にするためには、コネクター `DeliveryTransport` 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ーは、JMS トランSPORTを使用し、コネクター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SPORTを介して行われます。JMS トランSPORT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SPORT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、コネクターがイベントをパブリッシュして (コネクターが `pollForEvents()` メソッド内の `gotAppEvent()` 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。コネクターは再始動時に、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 回送信されることとなります。

保証付きイベント・デリバリー機能を、JMS イベント・ストアあり、またはなしで、JMS 対応コネクターのために構成することができます。

コネクタ・フレームワークがビジネス・オブジェクトを WebSphere InterChange Server Express 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、状況標識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。詳細については、55 ページの『イベント処理』を参照してください。

ビジネス・オブジェクト要求

ビジネス・オブジェクト要求は、InterChange Server Express が doVerbFor() メソッドにビジネス・オブジェクトを送信するときに処理されます。コネクタは、構成されているデータ・ハンドラーを使用してビジネス・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・ハンドラーについての要件を除いては、処理されるビジネス・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

コネクタは、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、各ビジネス・オブジェクトの動詞に基づいて処理します。サポートするビジネス・オブジェクトを処理するために、コネクタはビジネス・オブジェクト・ハンドラーと doForVerb() メソッドを使用します。コネクタは、以下のビジネス・オブジェクトの動詞をサポートします。

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

注: Create 動詞、Update 動詞、および Delete 動詞を持つビジネス・オブジェクトは、非同期的にも同期的にも送信できます。デフォルト・モードは非同期送信です。コネクタは、Retrieve 動詞、Exists 動詞、Retrieve by Content 動詞のビジネス・オブジェクトの非同期送信をサポートしません。したがって、Retrieve 動詞、Exists 動詞、または Retrieve by Content 動詞のデフォルト・モードは同期送信です。

Create、Update、および Delete

Create 動詞、Update 動詞、および Delete 動詞を持つビジネス・オブジェクトの処理は、ビジネス・オブジェクトが非同期的に送信されたか同期的に送信されたかによって決まります。

非同期デリバリー

これは、Create 動詞、Update 動詞、および Delete 動詞を持つビジネス・オブジェクトのデフォルト・デリバリー・モードです。データ・ハンドラーを使用して、ビジネス・オブジェクトからメッセージが作成され、出力キューに書き込まれます。メッセージがデリバリーされると、コネクタは SUCCESS を戻します。それ以外の場合は FAIL を戻します。

注: コネクタには、メッセージが受信されたかどうか、または、処置が行われたかどうかを確認する方法はありません。

同期デリバリー

コネクタ固有プロパティで `ReplyToQueue` が定義されており、かつビジネス・オブジェクトの変換プロパティに `responseTimeout` が存在する場合、コネクタは同期モードで要求を発行します。続いて、コネクタは、受信側のアプリケーションで適切な処置が行われたかどうかを確認するために応答を待ちます。

WebSphere MQ では、コネクタは次の表に示すようなヘッダーを持つメッセージを最初に発行します。

フィールド	説明	値
Format	フォーマット名	変換プロパティに定義されている出力フォーマット。IBM の要件に合わせて、8 文字を超える部分が切り捨てられます (例: MQSTR)。
MessageType	メッセージ・タイプ	MQMT_DATAGRAM* 受信側のアプリケーションからの応答を予期しない場合。 MQMT_REQUEST* 応答を予期する場合。
Report	要求されたレポート・メッセージのオプション	応答メッセージの返送が予測される場合、このフィールドには次の値が取り込まれます。処理が成功したときに肯定処理レポートが必要な場合は、MQRO_PAN*。処理が失敗したときに否定処理レポートが必要な場合は、MQRO_NAN*。生成されるレポートの相関 ID が最初に発行された要求のメッセージ ID と同じになる必要がある場合は、MQRO_COPY_MSG_ID_TO_CORREL_ID*。
ReplyToQueue	応答キューの名前	応答メッセージの返送が予測される場合、このフィールドにはコネクタ・プロパティ <code>ReplyToQueue</code> の値が取り込まれます。
Persistence	メッセージのパーシスタンス	MQPER_PERSISTENT*
Expiry	メッセージの存続時間	MQEI_UNLIMITED*

* は、IBM によって定義される定数を示します。

上記の表に示したメッセージ・ヘッダーの後に、メッセージの本体が続きます。メッセージの本体は、データ・ハンドラーを使用して直列化されたビジネス・オブジェクトです。

Report フィールドは、受信側アプリケーションから肯定処理レポートと否定処理レポートの両方の返送が予測されることを示すために設定されます。メッセージを発行したスレッドは、受信側アプリケーション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ちます。

コネクタから同期要求を受け取ると、アプリケーションはビジネス・オブジェクトを処理し、次の表に示すようなレポート・メッセージを発行します。

フィールド	説明	値
Format	フォーマット名	変換プロパティ内で定義された <code>busObj</code> の入力フォーマット
MessageType	メッセージ・タイプ	MQMT_REPORT*

* は、IBM によって定義される定数を示します。

動詞	Feedback フィールド	メッセージの本体
Create、Update、または Delete	SUCCESS VALCHANGE	(オプション) 変更を反映する、直列化されたビジネス・オブジェクト。
	VALDUPES FAIL	(オプション) エラー・メッセージ。

WebSphere MQ フィードバック・コード	等値の応答*
MQFB_NONE (フィードバック・コードが指定されない場合のデフォルトです)	VALCHANGE
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 (この応答後、コネクター・エージェントは即時に終了します)

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、feedback フィールドが MQFB_PAN (または特定の WebSphere Business Integration システムの値) に設定されたレポート・メッセージを作成します。また、オプションで、アプリケーションは、すべての変更を含む直列化されたビジネス・オブジェクトをメッセージ本体に取り込みます。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、feedback フィールドが MQFB_NAN (または特定の WebSphere Business Integration システムの値) に設定されたレポート・メッセージを作成します。オプションで、このレポート・メッセージの本体にエラー・メッセージを含むこともできます。いずれの場合も、アプリケーションはメッセージの correlationID フィールドをコネクター・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メッセージを送信します。

コネクターは、応答メッセージを検索すると、デフォルトでは、応答の correlationID を要求メッセージの messageID と突き合わせます。続いて、要求を発行したスレッドに通知を送信します。コネクターは、応答の feedback フィールドの設定によって、メッセージの本体にビジネス・オブジェクトとエラー・メッセージのどちらが含まれているかを予測します。フィードバック・コードが MQFB_NONE (フィードバック・コードが指定されない場合のデフォルト値) の場合、コネクターはデフォルトで戻りコードが VALCHANGE であると見なします。次に、コネクターは、応答メッセージをデータ・ハンドラーに渡し、ビジネス・オブジェクトを更新します。ビジネス・オブジェクトが含まれていると予測したにもかかわらず、メッセージの本体にビジネス・オブジェクトが取り込まれていなかった場合、コネクターは InterChange Server Express が Request 操作のために最初に発行したのと同

ビジネス・オブジェクトを単純に返送します。エラー・メッセージが含まれていると予測したにもかかわらず、メッセージの本体にエラー・メッセージが取り込まれていなかった場合、InterChange Server Express には応答コードと汎用エラー・メッセージが返送されます。ただし、メッセージ選択子を使用して、識別やフィルター操作を行うこともできます。あるいは、アダプターが特定の要求に対して応答メッセージを識別する方法を制御できます。このメッセージ選択子機能は、JMS 機能です。この機能は同期要求処理にのみ摘要されます。以下に詳細を説明します。

メッセージ選択子を使用した応答メッセージのフィルター操作: コネクターは、同期要求処理用のビジネス・オブジェクトを受け取ると、動詞のアプリケーション固有情報に `response_selector` スtringが含まれていないかどうかをチェックします。`response_selector` が未定義の場合、コネクターは、前述のように、相関 ID を使用して応答メッセージを識別します。

`response_selector` が定義されていると、コネクターは次の構文に基づく名前 - 値のペアを探します。

```
response_selector=JMSCorrelationID LIKE'selectorstring'
```

メッセージ選択子Stringは、応答を一意的に識別する必要があります。また、次の例に示すように、値は単一引用符で囲む必要があります。

```
response_selector=JMSCorrelationID LIKE 'Oshkosh'
```

上記の例の場合、アダプターは、要求メッセージを発行した後、「Oshkosh」に等しい相関 ID を持つ応答メッセージの `ReplyToQueue` をモニターします。アダプターは、このメッセージ選択子に一致する最初のメッセージを検索し、応答としてディスパッチします。

また、オプションで、アダプターによる実行時置換を実行して、各要求ごとに固有のメッセージ選択子を生成することもできます。メッセージ選択子の代わりに、'`{1}`' のように、整数を中括弧で囲んだ形式でプレースホルダーを指定します。この後にコロンを記入し、置換に使用する属性をコンマで区切ってリストします。プレースホルダー内の整数は、置換時に使用される属性のインデックスとして機能します。次のメッセージ選択子を例に考えてみます。

```
response_selector=JMSCorrelationID LIKE '{1}': MyDynamicMO.CorrelationID
```

このメッセージ選択子は、アダプター `{1}` を選択子に続く最初の属性の値 (この例では、子オブジェクト `MyDynamicMO` の属性 `CorrelationId`) と置換するように通知します。属性 `CorrelationID` の値が `123ABC` である場合には、アダプターは以下の基準によって作成されたメッセージ選択子を生成し、使用します。

```
JMSCorrelation LIKE '123ABC'
```

これで、応答メッセージが識別されます。

また、次のように、複数の置換を指定することも可能です。

```
response_selector=PrimaryId LIKE '{1}' AND AddressId LIKE '{2}' :  
PrimaryId, Address[4].AddressId
```

この例では、アダプターは {1} をトップレベル・ビジネス・オブジェクトの属性 PrimaryId の値で置換し、{2} を子コンテナ・オブジェクト Address の 5 番目の位置にある AddressId の値で置換します。この方法により、応答メッセージ選択子に指定されたビジネス・オブジェクトおよびメタオブジェクト内の、すべての属性を参照することができます。Address[4].AddressId を使用した詳細検索の実行方法について詳しくは、「JCDK API マニュアル」(getAttribute メソッド) を参照してください。

次のいずれかの状況が発生すると、実行時にエラーが報告されます。

- '{}' シンボルの間に整数以外の値を指定した場合
- 属性が定義されていないインデックスを指定した場合
- 指定された属性がビジネス・オブジェクトまたはメタオブジェクトに存在しない場合
- 属性パスの構文が不正の場合

例えば、メッセージ選択子にリテラル値「{」または「}」を組み込む場合には、それぞれ「{{」または「}}」を使用できます。また、属性値にこれらの文字を組み込むこともできますが、その場合、最初の「{」は不要です。エスケープ文字を使用した次の例について考えてみます。response_selector=JMSCorrelation LIKE '{1}' and CompanyName='A{P: MyDynamicMO.CorrelationID

コネクターはこのメッセージ選択子を次のように解決します。

```
JMSCorrelationID LIKE '123ABC' and CompanyName='A{P'
```

コネクターが属性値内で検出した特殊文字（「{」、「}」、「:」、または「;」など）は、照会ストリングに直接挿入されます。このため、アプリケーション固有情報の区切り文字としても機能する特殊文字を、照会ストリングに組み込むことができます。

次の例は、リテラル・ストリングの置換値が属性値から抽出される方法を示しています。

```
response_selector=JMSCorrelation LIKE '{1}' and CompanyName='A{P':  
MyDynamicMO.CorrelationID
```

MyDynamicMO.CorrelationID に値 {A:B}C;D が含まれていると、コネクターはメッセージ選択子を次のように解決します。 JMSCorrelationID LIKE '{A:B}C;D' and CompanyName='A{P'

応答選択子コードについて詳しくは、JMS 1.0.1 の仕様書を参照してください。

双方向言語サポート: Adapter for WebSphere MQ では、ビジネス・オブジェクト・コンテンツの双方向変換をサポートしています。双方向言語サポートは、コネクターが外部の WebSphere MQ アプリケーションとメタデータを交換し、同期デリバリー・フィルターの仕組みを使用して ReplyToQueue で応答メッセージの場所を探索する場合にも提供されます。こうした環境では、双方向変換を適用することにより、Windows 標準形式を WebSphere MQ アプリケーションの双方向形式に変換します。この処理は、BiDi.Application 標準構成プロパティに設定された属性を使用して実行されます。

次の構文規則に従って response_selector が指定されているとします。

```
response_selector=JMSCorrelationID LIKE'selectorstring'
```

また、双方向文字が「*selectorstring*」に表示されている場合、アダプターを構成して「*selectorstring*」を外部アプリケーションの双方向形式に変換できます。外部アプリケーションは、アダプターによって作成される要求に対する応答メッセージを送信する役割を持つアプリケーションです。この動作は、外部アプリケーションの双方向形式が、Connector Designer アプリケーションで最初に定義されたアプリケーション固有情報のデフォルト Windows 形式と異なる場合に必要になります。そのような場合は、BiDi.Application 標準構成プロパティを外部アプリケーションの双方向形式によって構成する必要があり、その後アダプターは、応答メッセージのフィルタリングで使用する前に、メッセージ・セレクター・ストリングを変換します。

カスタム・フィードバック・コードの作成: コネクター・プロパティ

FeedbackCodeMappingMO を指定することにより、WebSphere MQ フィードバック・コードを拡張してデフォルトの解釈をオーバーライドすることができます。このプロパティを使用すると、WebSphere Business Integration システム固有のすべての戻り状況値を WebSphere MQ フィードバック・コードにマップしたメタオブジェクトを作成できます。(メタオブジェクトを使用して) フィードバック・コードに割り当てられた戻り状況値は、InterChange Server Express に渡されます。詳細については、27 ページの『FeedbackCodeMappingMO』を参照してください。

Retrieve、Exists、および Retrieve By Content

Retrieve 動詞、Exists 動詞、および Retrieve By Content 動詞を持つビジネス・オブジェクトは、同期送信のみをサポートします。コネクターは、これらの動詞を持つビジネス・オブジェクトを、Create 動詞、Update 動詞、および Delete 動詞に対して定義されている同期送信と同様に処理します。ただし、Retrieve 動詞、Exists 動詞、および Retrieve By Content 動詞を使用する場合には、responseTimeout と replyToQueue が必須です。さらに、Retrieve By Content 動詞と Retrieve 動詞の場合、トランザクションを完了するためにはメッセージの本体に直列化されたビジネス・オブジェクトが取り込まれている必要があります。

次の表に、これらの動詞に対応する応答メッセージを示します。

動詞	Feedback フィールド	メッセージの本体
Retrieve または RetrieveByContent	FAIL FAIL_RETRIEVE_BY_CONTENT	(オプション) エラー・メッセージ。
	MULTIPLE_HITS SUCCESS	直列化されたビジネス・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

インストールが完了したコネクターを始動する前に、コネクターを構成する必要があります。このセクションでは、ほとんどの開発者が実行する必要のある、構成と始動に関するいくつかの作業について概要を説明します。

アダプターのインストール

何をどこにインストールするかについての詳細は、21 ページの『第 2 章 アダプター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。一部のプロパティはデフォルト値を持っており、変更を加えなくても使用できます。また、一部のプロパティについては、コネクタを実行する前に値を設定する必要があります。詳細については、21 ページの『第 2 章 アダプターのインストールおよび構成』を参照してください。

Adapter for WebSphere MQ のコネクタ・プロパティを構成する際には、次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、使用している WebSphere MQ サーバーのホストの対応する値に一致している。
- コネクタ・プロパティ Port に指定した値が、使用しているキュー・マネージャーのリスナーのポートの対応する値に一致している。
- コネクタ・プロパティ Channel に指定した値が、使用しているキュー・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。
- コネクタ・プロパティ InputQueue、InProgressQueue、ArchiveQueue、ErrorQueue、および UnsubscribeQueue のキュー URI が有効であり、実際に存在する。

通知なしで要求を送信するためのコネクタの構成

通知なしで要求を送信 (デフォルト非同期モード、別名「fire and forget」) するようにコネクタを構成するには、次の作業を実行します。

- 送信する要求を表しており、コネクタ向けに構成したデータ・ハンドラーとの互換性もあるビジネス・オブジェクトを作成します。
- 静的メタオブジェクトまたは動的メタオブジェクトを使用して、宛先の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の詳細については、39 ページの『静的メタオブジェクトの作成の概要』と 41 ページの『動的子メタオブジェクトの作成の概要』を参照してください。
- (静的または動的) メタオブジェクト内のプロパティ ResponseTimeout を -1 に設定します。この設定では、コネクタは発行したビジネス・オブジェクトの戻りをチェックしません。
- 詳細については、11 ページの『Create、Update、および Delete』、33 ページの『メタオブジェクトの構成』、および 49 ページの『第 3 章 ビジネス・オブジェクトの作成および変更』を参照してください。

要求を送信して通知を取得するためのコネクタの構成

要求を送信して通知を取得 (同期イベント処理) するようにコネクタを構成するには、次の作業を実行します。

- コネクタが応答を待機する時間を指示するために正の ResponseTimeout 値を指定する点を除いて、『通知なしで要求を送信するためのコネクタの構成』の説明にある手順に従います。

- コネクターが予期する応答メッセージの具体的な詳細については、11 ページの『Create、Update、および Delete』を参照してください。示されている要件を応答メッセージが満たしていない場合、コネクターはエラーを報告したり、応答メッセージを認識できなかったりする可能性があります。33 ページの『メタオブジェクトの構成』と49 ページの『第 3 章 ビジネス・オブジェクトの作成および変更』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は、ユーザーがビジネス・オブジェクトに関して指定したアプリケーション固有の情報と、コネクターによるビジネス・オブジェクトの処理方法についての情報を格納します。静的メタオブジェクトは、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すべての情報を、コネクターの始動時に提供します。

さまざまな種類のビジネス・オブジェクトの送信先であるキューが実装時にわかっている場合は、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および構成するには、次の作業を実行します。

- 39 ページの『静的メタオブジェクトの作成の概要』の手順に従います。
- コネクター固有のプロパティ `ConfigurationMetaObject` 内で静的メタオブジェクトの名前を指定することにより、コネクターが静的メタオブジェクトにサブスクライブするようにします。詳細については、24 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異なるビジネス・オブジェクト処理を実行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は、要求の処理方法をコネクターに (実行時に) 指示します。静的メタオブジェクトは、コネクターがビジネス・オブジェクトを処理するために必要なすべての情報を、コネクターに提供します。これと異なり、動的メタオブジェクトは、特定のシナリオの処理を実行するために必要なロジックの追加部分だけを提供します。動的メタオブジェクトを作成および構成するには、次の作業を実行します。

- 動的メタオブジェクトを作成し、それを子オブジェクトとして要求ビジネス・オブジェクトに追加します。
- コラボレーションのプログラムに、動的メタオブジェクトをコネクターに対して発行する前に、宛先キューやメッセージ・フォーマットなどの情報をそのメタオブジェクトに取り込むロジックを追加します。

コネクターは動的メタオブジェクトをチェックし、その情報を使用してビジネス・オブジェクトの処理方法を判別します。詳細については、41 ページの『動的子メタオブジェクトの作成の概要』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、メッセージがアプリケーション間で送信されるときにアプリケーション・データに添付される制御情報が格納されます。静的メタオブジェクトまたは動的メタオブジェクト内で、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、11 ページの『Create、Update、および Delete』を参照してください。

キュー URI の構成

WebSphere MQ 用のアダプターと共に使用するキューを構成するには、次の作業を実行します。

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

```
queue://<キュー・マネージャー名>/<実際のキュー>
```

- コネクター固有の構成プロパティに、キュー・マネージャーのホストを指定します。
- ターゲット・アプリケーションが MQMD ヘッダーのみを予期していて、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、?targetClient=1 をキュー URI に付加します。詳細については、32 ページの『キューの Uniform Resource Identifier (URI)』と WebSphere MQ のプログラミング・ガイドを参照してください。

データ・ハンドラーの構成

データ・ハンドラーを構成するには、次の 2 つの方法があります。

- コネクター固有のプロパティ DataHandlerClassName 内で、データ・ハンドラー・クラス名を指定します。詳細については、24 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。
- または、コネクター固有のプロパティ DataHandlerMimeType および DataHandlerConfigMO 内で、MIME タイプとその MIME タイプの構成を定義するデータ・ハンドラー・メタオブジェクトをそれぞれ指定します。詳細については、「データ・ハンドラー・ガイド」を参照してください。

始動スクリプトの変更

コネクターの始動方法の詳細については、21 ページの『第 2 章 アダプター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、コネクター・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・ライブラリーの格納場所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 サーバーに対し、最新で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、46 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 アダプターのインストールおよび構成

- 『インストール作業の概要』
- 『アダプターおよび関連ファイルのインストール』
- 『インストール済みファイルの構造』
- 23 ページの『コネクタ構成』
- 32 ページの『キューの Uniform Resource Identifier (URI)』
- 30 ページの『複数コネクタ・インスタンスの作成』
- 33 ページの『メタオブジェクトの構成』
- 46 ページの『始動ファイルの構成』
- 46 ページの『コネクタの始動』
- 47 ページの『コネクタの停止』

この章では、コネクタのインストール方法および構成方法と、メッセージ・フローをコネクタとともに動作させるための構成方法について説明します。

インストール作業の概要

WebSphere MQ 対応コネクタをインストールするには、以下の作業を行う必要があります。

- **統合ブローカーのインストール** この作業では、WebSphere Business Integration システムのインストールと統合ブローカーの始動を行います。作業の詳細については、使用するブローカーおよびオペレーティング・システムのインストール文書に説明があります。
- **アダプターおよび関連ファイルのインストール** この作業では、アダプターのファイルをソフトウェア・パッケージから使用システムにインストールします。『アダプターおよび関連ファイルのインストール』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 Server Express アダプター製品のインストールの詳細については、「WebSphere Business Integration Server Express インストール・ガイド」(Windows 版、Linux 版、OS/400 および i5/OS 版)を参照してください。このガイドは、WebSphere Business Integration Server Express Adapters Infocenter (<http://www.ibm.com/websphere/wbiserverexpress/infocenter>) にあります。

インストール済みファイルの構造

以下のセクションでは、インストール後の製品のパスとファイル名について説明します。

注: Windows 環境でも Linux 環境でも、一般に、WebSphere MQ と JMS は異なるディレクトリーにインストールされています。例えば Linux システムの場合、デフォルトでは、WebSphere MQ は /var/mqm/ にインストールされ、JMS は

/usr/mqm/java/lib にインストールされています。JMS のインストールを /var/mqm/java/lib にリダイレクトすることにより、ルーチン /usr に関連するシステム管理タスクによる削除を防止することができます。同様に Windows でも、通常、WebSphere MQ は %Program Files%WebSphere MQ の下にインストールされ、JMS は %Program Files%IBM%MQSeries%Java の下にインストールされます。これに従って、WebSphere MQ コネクタースタートスクリプト内のクラスパスを変更してください。

Windows のファイル構造

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

ユーティリティーにより、コネクタが *ProductDir*\connectors\WebSphereMQ ディレクトリーにインストールされ、コネクタへのショートカットが「スタート」メニューに追加されます。

次の表に、コネクタが使用する Windows ファイル構造の説明と、インストーラーからコネクタのインストールを選択した場合に自動的にインストールされるファイルを示します。

<i>ProductDir</i> のサブディレクトリー	説明
connectors\WebSphereMQ\CWWebSphereMQ.jar	WebSphere MQ コネクタによってのみ使用されるクラスを含む
connectors\WebSphereMQ\start_WebSphereMQ.bat	コネクタの始動スクリプト
connectors\WebSphereMQ\start_WebSphereMQ_service.bat	コネクタ・サービスの始動スクリプト。
connectors\messages\WebSphereMQConnector.txt	コネクタのメッセージ・ファイル
bin\Data\App\WebSphereMQConnectorTemplate	アダプター定義のテンプレート・ファイル

注: 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。

Linux のファイル構造

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

ユーティリティーにより、コネクタが *ProductDir*/connectors/WebSphereMQ ディレクトリーにインストールされます。

次の表に、コネクタが使用する Linux ファイル構造の説明と、インストーラーからコネクタのインストールを選択した場合に自動的にインストールされるファイルを示します。

<i>ProductDir</i> のサブディレクトリー	説明
connectors/WebSphereMQ/CWWebSphere MQ.jar	WebSphere MQ コネクタによってのみ使用されるクラスを含む

<i>ProductDir</i> のサブディレクトリー	説明
connectors/WebSphereMQ/start_WebSphereMQ.sh	コネクターのシステム始動スクリプト。このスクリプトは、汎用のコネクター・マネージャー・スクリプトから呼び出されます。
connectors/messages/WebSphereMQ/Connector.txt	System Manager の「コネクター構成」画面をクリックすると、このコネクター・マネージャー・スクリプトのカスタマイズ済みラッパーがインストーラーによって作成されます。コネクターの始動および停止には、このカスタマイズされたラッパーを使用してください。
bin/Data/App/WebSphereMQConnectorTemplate	コネクターのメッセージ・ファイル アダプター定義のテンプレート・ファイル

注: 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。

i5/OS のファイル構造

デフォルトでは、アダプターは /QIBM/ProdData/WBIServer44/product ディレクトリー (ProductDir と呼ぶ) にインストールされます。表 3 に、コネクターが使用する i5/OS ファイル構造と、標準インストール時にインストールされるファイルを示します。

<i>ProductDir</i> のサブディレクトリー	説明
connectors/WebSphereMQ	CWWebSphereMQ.jar ファイルおよび start_WebSphereMQ.sh 始動スクリプトが格納されています。
connectors/messages	WebSphereMQconnector.txt メッセージ・ファイルおよび WebSphereMQConnector_II_TT.txt ファイル (言語 (II) および国または地域 (TT) に固有のメッセージ・ファイル) が格納されています。
lib	WBIA.jar ファイルが格納されています。
bin	CWConnEnv.sh ファイルが格納されています。

WebSphere Business Integration Console 機能を使用して、コネクターをすばやく始動することができます。詳しくは、Console に付属のオンライン・ヘルプを参照してください。

コネクター構成

コネクターの構成プロパティーには、標準構成プロパティーとアダプター固有の構成プロパティーという 2 つのタイプがあります。アダプターを実行する前に、これらのプロパティーの値を設定する必要があります。

コネクターのプロパティーを構成するには、Connector Configurator Express を使用します。

- Connector Configurator Express の説明と段階的な手順については、83 ページの『付録 B. Connector Configurator』を参照してください。
- 標準コネクター・プロパティーの説明については、24 ページの『標準コネクター・プロパティー』、および 57 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

- コネクター固有のプロパティの詳細については、『コネクター固有のプロパティ』を参照してください。

コネクターは、始動時に構成値を取得します。実行時セッション中に、1 つ以上のコネクター・プロパティの値の変更が必要になることがあります。

AgentTraceLevel など一部のコネクター構成プロパティへの変更は、即時に有効になります。その他のコネクター・プロパティへの変更を有効にするには、変更後にコンポーネントまたはシステムを再始動する必要があります。あるプロパティが動的 (即時に有効になる) か静的 (コネクター・コンポーネントまたはシステムを再始動する必要がある) かを判別するには、Connector Configurator Express の「コネクター・プロパティ」ウィンドウ内の「更新メソッド」列を参照してください。

標準コネクター・プロパティ

標準構成プロパティにより、すべてのコネクターによって使用される情報が提供されます。標準構成プロパティの資料については、57 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。

注: Connector Configurator Express で構成プロパティを設定するときは、BrokerType プロパティで使用するブローカーを指定します。このプロパティの値を設定すると、使用するブローカーに関連するプロパティが「Connector Configurator Express」ウィンドウに表示されます。

コネクター固有のプロパティ

コネクター固有の構成プロパティには、コネクターが実行時に必要とする情報が用意されています。コネクター固有の構成プロパティは、エージェントを再コーディングまたは再ビルドせずに、コネクター内部の静的情報またはロジックを変更する手段にもなっています。

次の表に、アダプターのコネクター固有構成プロパティを示します。プロパティの説明については、以下の各セクションを参照してください。

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キューが送信されるキュー	queue://crossworlds.queue.manager/MQCONN.ARCHIVE	いいえ
CCSID	キュー・マネージャーの接続に使用する文字セット		いいえ
Channel	MQ サーバー・コネクター・チャネル		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・クラス名	com.crossworlds.DataHandlers.text.xml	いいえ
DataHandlerConfigMO	データ・ハンドラー・メタオブジェクト	MO_DataHandler_Default	はい
DataHandlerMimeType	MIME ファイル・タイプ	text/xml	いいえ
DefaultVerb	コネクターがサポートする任意の動詞		いいえ
EnableMessageProducerCache	true または false	true	いいえ

名前	指定可能な値	デフォルト値	必須
ErrorQueue	未処理のメッセージ・キュー	queue://crossworlds. queue.manager/MQCONN.ERROR	いいえ
FeedbackCodeMappingMO	フィードバック・コード・メタオブジェクト		いいえ
HostName	WebSphere MQ サーバー		はい
InDoubtEvents	FailOnStartup Reprocess IgnoreLogError	Reprocess	いいえ
InputQueue	ポーリング・キュー	queue://crossworlds. queue.manager/MQCONN.IN	いいえ
InProgressQueue	進行中のイベント・キュー	queue://crossworlds.queue. manager/MQCONN.IN_PROGRESS	いいえ
PollQuantity	<i>InputQueue</i> プロパティ内で指定された各キューから検索されるメッセージの数	1	いいえ
Port	WebSphere MQ リスナー用に設定されたポート		はい
ReplyToQueue	コネクターからの要求発行時に応答メッセージが配信されるキュー	queue://crossworlds. queue.manager/MQCONN.REPLY	いいえ
ReplyToQueuePollFrequency	同期要求処理時における受信側のポーリング間隔 (ミリ秒単位)		いいえ
SessionPoolSizeForRequests	要求処理中に使用されるセッションをキャッシュする最大プール・サイズ	10	いいえ
UnsubscribedQueue	アンサブスクライブされたメッセージが送信されるキュー	queue://crossworlds.queue. manager/MQCONN.UNSUBSCRIBED	いいえ
UseDefaults	true または false	false	いいえ

ApplicationPassword

WebSphere MQ にログインするために、UserID とともに使用されるパスワードです。

デフォルト = 設定値なし

ApplicationPassword の値がブランクのままか、または除去された場合、コネクターは WebSphere MQ によって提供されるデフォルトのパスワードを使用します。

注*

ApplicationUserName

WebSphere MQ にログインするために、Password とともに使用されるユーザー ID です。

デフォルト = 設定値なし

ApplicationUserName の値がブランクのままか、または除去された場合、コネクターは WebSphere MQ によって提供されるデフォルトのユーザー ID を使用します。

注*

ArchiveQueue

正常に処理されたメッセージのコピーが送信されるキューです。

デフォルト = `queue://crossworlds.queue.manager/MQCONN.ARCHIVE`

CCSID

キュー・マネージャーの接続に使用する文字セット。このプロパティーの値は、キュー URI 内の CCSID プロパティーの値と一致する必要があります。32 ページの『キューの Uniform Resource Identifier (URI)』を参照してください。

デフォルト = 設定値なし

Channel

コネクタが WebSphere MQ と通信するとき使用する MQ サーバー・コネクタ・チャンネルです。

デフォルト = 設定値なし

Channel の値がブランクのままか、または除去された場合、コネクタは WebSphere MQ によって提供されるデフォルトのサーバー・チャンネルを使用しません。注*

ConfigurationMetaObject

コネクタの構成情報を含む静的なメタオブジェクトの名前です。

デフォルト = 設定値なし

DataHandlerClassName

ビジネス・オブジェクトとの間でのメッセージ変換に使用するデータ・ハンドラー・クラスです。

デフォルト = `com.crossworlds.DataHandlers.text.xml`

DataHandlerConfigMO

構成情報を提供するために、データ・ハンドラーに渡されるメタオブジェクト。

デフォルト = `MO_DataHandler_Default`

DataHandlerMimeType

使用すると、特定の MIME タイプに基づいたデータ・ハンドラーを要求できます。

デフォルト = `text/xml`

DefaultVerb

着信ビジネス・オブジェクト内に設定する動詞を指定します。ただし、この動詞がポーリング中にデータ・ハンドラーにより設定されていないことが前提です。

デフォルト = 設定値なし

EnableMessageProducerCache

要求メッセージを送信するために、アダプターがメッセージ・プロデューサーのキャッシュを有効にすることを指定する boolean プロパティ。

デフォルト= true

ErrorQueue

処理されなかったメッセージが送信されるキューです。

デフォルト = queue://crossworlds.queue.manager/MQCONN.ERROR

FeedbackCodeMappingMO

メッセージの受信を InterChange Server Express に同期的に確認するために使用され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て再割り当てするプロパティです。このプロパティを使用すると、フィードバック・コードを表示するために各属性名を解釈するときの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 Express に渡される戻り状況値です。デフォルトのフィードバック・コードのリストは、12 ページの『同期デリバリー』を参照してください。コネクタは、WebSphere 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入れます。

- MQFB_APPL_FIRST
- MQFB_APPL_FIRST_OFFSET_N、N は整数 (MQFB_APPL_FIRST + N の値として解釈される)
- MQFB_NONE
- MQFB_PAN
- MQFB_NAN

コネクタは、以下の WebSphere Business Integration システム固有の状況コードを、メタオブジェクトの属性値として受け入れます。

- 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。

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 = 設定値なし

HostName

WebSphere MQ をホスティングしているサーバーの名前です。

デフォルト = 設定値なし

InDoubtEvents

コネクターの予期しないシャットダウンのために、処理が完了していない進行中イベントの処理方法を指定します。初期化中に進行中のキューにイベントが見つかった場合に実行するアクションを、以下の 4 つから選択してください。

- **FailOnStartup。** エラーを記録し、即時にシャットダウンします。
- **Reprocess。** 残っているイベントを最初に処理し、続いて入力キュー内のメッセージを処理します。
- **Ignore。** 実行中のキューに残っているすべてのメッセージを破棄します。
- **LogError。** エラーを記録しますが、シャットダウンはしません。

デフォルト = Reprocess

InputQueue

コネクターが新規のメッセージの有無を確認するためにポーリングするメッセージ・キューです。コネクターは、セミコロンで区切られた複数のキュー名を受け入れます。例えば、MyQueueA、MyQueueB、および MyQueueC の 3 つのキューにポーリングするには、コネクター構成プロパティ *InputQueue* の値を MyQueueA;MyQueueB;MyQueueC とします。

InputQueue プロパティが指定されていない場合、コネクターは正常に始動して警告メッセージを印刷し、要求処理のみを実行します。この場合はイベント処理は実行しません。

コネクターはラウンドロビン方式でキューをポーリングし、各キューから *pollQuantity* で指定された値を最大数とするメッセージを検索します。例えば、*pollQuantity* が 2 であり、MyQueueA に 2 件のメッセージがあり、MyQueueB に 1 件のメッセージがあり、MyQueueC に 5 件のメッセージがある場合は、コネクターは以下のようにメッセージを取得します。

PollQuantity が 2 に設定されているため、コネクターは、*pollForEvents* への 1 回の呼び出しごとに各キューからそれぞれ最大 2 つのメッセージを検索します。最初のサイクル (2 回のうちの 1 回目) では、コネクターは、MyQueueA、MyQueueB、および MyQueueC の各キューの 1 番目のメッセージを検索します。これによって、ポーリングの第 1 ラウンドが完了します。*PollQuantity* が 1 に設定されている場合、コネクターはこの時点で停止します。この例では *PollQuantity* が 2 に設定されているため、コネクターは第 2 ラウンド (2/2 ラウンド) のポーリングを開始し、MyQueueA と MyQueueC の各キューからそれぞれ 1 つずつのメッセージを検索します。このとき、MyQueueB は空になっているためスキップされます。すべてのキューを 2 回ずつポーリングしたら、メソッド *pollForEvents* への呼び出しは完了します。以下に、メッセージ検索の順序を示します。

1. MyQueueA から 1 件のメッセージ
2. MyQueueB から 1 件のメッセージ
3. MyQueueC から 1 件のメッセージ

4. MyQueueA から 1 件のメッセージ
5. 空になったため、MyQueueB をスキップ
6. MyQueueC から 1 件のメッセージ

デフォルト = `queue://crossworlds.queue.manager/MQCONN.IN`

InProgressQueue

処理中にメッセージが保留されるメッセージ・キューです。System Manager を使用してデフォルトの InProgressQueue 名をコネクタ固有のプロパティから除去することにより、このキューなしで動作するようにコネクタを構成できます。このようにすると、始動時にイベントが保留されているときにコネクタをシャットダウンするとイベント・デリバリーで問題が発生する可能性があることを示す警告メッセージが出されます。

デフォルト = `queue://crossworlds.queue.manager/MQCONN.IN_PROGRESS`

PollQuantity

pollForEvents スキャン中に、InputQueue プロパティで指定した各キューから取得するメッセージの数です。

デフォルト = 1

Port

WebSphere MQ リスナー用に設定されたポートです。

デフォルト = 設定値なし

ReplyToQueue

コネクタからの要求発行時に応答メッセージが配信されるキューです。子動的メタオブジェクトの属性を使用して応答を無視することもできます。このような属性の詳細については、43 ページの『JMS ヘッダーと動的子メタオブジェクトの属性』を参照してください。

デフォルト = `queue://crossworlds.queue.manager/MQCONN.REPLY`

ReplyToQueuePollFrequency

同期要求処理時における受信側のポーリング間隔を指定します。この値はミリ秒単位です。

デフォルト = 設定値なし

SessionPoolSizeForRequests

要求処理中に使用されるセッションをキャッシュする最大プール・サイズ。

デフォルト = 10

UnsubscribedQueue

アンサブスクライブされたメッセージが送信されるキューです。

デフォルト = `queue://crossworlds.queue.manager/MQCONN.UNSUBSCRIBED`

注: * WebSphere MQ によって提供される値は誤っていたり不明である可能性があるため、常にチェックする必要があります。値が誤っていたり不明な場合は、値を暗黙的に指定してください。

UseDefaults

Create 操作の場合、UseDefaults を true に設定すると、コネクタは、各 isRequired ビジネス・オブジェクト属性に有効値またはデフォルト値が指定されているかどうかをチェックします。値が指定されている場合、Create 操作は成功します。このパラメータを false に設定すると、コネクタは有効値の有無だけをチェックし、有効値が指定されていない場合、Create 操作は失敗します。デフォルトは false です。

複数コネクタ・インスタンスの作成

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

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

新規ディレクトリの作成

- **Windows** プラットフォームの場合:

```
ProductDir¥connectors¥connectorInstance
```

コネクタに、コネクタ固有のメタオブジェクトがある場合は、コネクタ・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、次のディレクトリを作成して、ファイルをそこに格納します。

```
ProductDir¥repository¥connectorInstance
```

ここで、*connectorInstance* によりコネクタ・インスタンスを一意に識別できません。

InterChange Server Express サーバー名は、例えば start_WEBSPHEREMQ.bat connName serverName のように、startup.bat のパラメータとして指定できます。

- **i5/OS** プラットフォームの場合:

```
/QIBM/UserData/WBIServer44/WebSphereICSName  
/connectors/connectorInstance
```

ここで、*connectorInstance* はコネクタ・インスタンスを一意的に示し、*WebSphereICSName* はコネクタの実行に使用する Interchange Server Express インスタンスの名前です。

コネクタに、コネクタ固有のメタオブジェクトがある場合は、コネクタ・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、次のディレクトリを作成して、ファイルをそこに格納します。

```
/QIBM/UserData/WBIServer44/WebSphereICSName
```

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

- **Linux プラットフォームの場合:**

ProductDir/connectors/connectorInstance。ここで connectorInstance は、コネクタ・インスタンスを一意的に示します。コネクタに、コネクタ固有のメタオブジェクトがある場合は、コネクタ・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、ディレクトリ ProductDir/repository/connectorInstance を作成し、ファイルをここに格納します。InterChange Server Express のサーバー名を connector_manager のパラメータとして指定することができます。例: connector_manager -start connName WebSphereICSName [-cConfigFile]

ビジネス・オブジェクト定義の作成

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

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

```
ProductDir¥repository¥initialConnectorInstance
```

追加作成したファイルは、ProductDir¥repository の適切な connectorInstance サブディレクトリに保管する必要があります。

コネクタ定義の作成

Connector Configurator Express 内で、コネクタ・インスタンスの構成ファイル (コネクタ定義) 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. 初期コネクタの構成ファイル (コネクタ定義) をコピーし、ファイルを名前変更する。
2. 各コネクタ・インスタンスがサポートしているビジネス・オブジェクト (および関連メタオブジェクト) が正しくリストされていることを確認する。
3. 適宜コネクタ・プロパティをカスタマイズする。

始動スクリプトの作成

始動スクリプトを作成するには、次の作業を実行します。

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

dirname

- この始動スクリプトを 31 ページの『ビジネス・オブジェクト定義の作成』で作成したコネクタ・ディレクトリーに置く。
- (Windows の場合のみ) 始動スクリプトのショートカットを作成する。
- (Windows の場合のみ) 初期コネクタのショートカット・テキストをコピーして、コマンド行から新規コネクタ・インスタンスに合うように初期コネクタ名を変更する。
- (i5/OS の場合のみ) 次の情報を使用して、コネクタのジョブ記述を作成します。
CRTDUPOBJ (QWBIWEBMQC) FROMLIB(QWBISVR44)OBJTYPE(*JOB)TOLIB (QWBISVR44) NEWOBJ(newwebspheremqname)。ここで、newwebspheremqname は新規コネクタのジョブ記述で使用する 10 文字の名前です。
- (i5/OS の場合のみ) 新規コネクタを WebSphere Business Integration Server Express Console に追加します。WebSphere Business Integration Server Express Console の詳細については、コンソールに付属のオンライン・ヘルプを参照してください。

キューの Uniform Resource Identifier (URI)

キューの URI は、シーケンス `queue://` で始まり、それに続いて以下のものが記述されます。

- キューが存在しているキュー・マネージャーの名前
- 別の /
- キューの名前
- (オプション) 残りのキュー・プロパティの、名前と値のペアのリスト

例えば、次の URI を指定した場合、キュー・マネージャー `crossworlds.queue.manager` 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://crossworlds.queue.manager/MQCONN.IN?targetClient=1&priority=5
```

以下の表に、キュー URI のプロパティ名を示します。

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 = 無制限。正の整数 = タイムアウト (ミリ秒単位)。
priority	メッセージの優先順位	0 から 9 で、1 が最高の優先順位。値 -1 は、このプロパティがキューの構成によって決定されることを意味します。値 -2 は、コネクタ自身のデフォルト値を使用できるように指定します。

プロパティ名	説明	値
persistence	メッセージをディスクに「ハード化」するかどうか	1 = 非永続 2 = 永続値 -1 は、このプロパティがキューの構成によって決定されることを意味します。値 -2 は、コネクタ自身のデフォルト値を使用できるように指定します。
CCSID	アウトバウンド・メッセージをエンコードする文字セット	整数: WebSphere MQ の資料にリストされている有効な値。この値は、CCSID コネクタ固有の構成プロパティの値と一致する必要があります。26 ページの『CCSID』を参照してください。
targetClient	受信側アプリケーションが JMS 準拠であるかどうか	0 = JMS (MQRFH2 ヘッダー) 1 = MQ (MQMD ヘッダーのみ)
encoding	数値フィールドの表示方法	基本的な WebSphere MQ 資料に記載されている整数値。

注: アダプターは、MQMessage 内のデータの文字セット (CCSID) またはエンコード属性を制御できません。データ変換はデータがメッセージ・バッファから検索されるかメッセージ・バッファにデリバリーされるかに行われるため、コネクタは JMS の IBM WebSphere MQ インプリメンテーションに依存してデータ変換を行います (IBM WebSphere MQ Java クライアント・ライブラリの資料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。コネクタは、変換プロセスにおける差異または失敗を制御できません。コネクタは、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 またはエンコードのメッセージを送信するには、出力キューが完全修飾 URI であり、CCSID および encoding の値を指定している必要があります。コネクタはこの情報を 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。多くの場合、CCSID およびエンコードのサポートの欠如は、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によって解決できます。それでも CCSID およびエンコードに関する問題が解消されない場合は、WebSphere Business Integration システム・テクニカル・サポートに連絡し、代替の Java Virtual Machine を使用してコネクタを実行することを検討してください。

メタオブジェクトの構成

コネクタは、メタオブジェクト・エントリを使用して、メッセージに関連付けるビジネス・オブジェクトを決定します。イベント・メッセージの処理に使用されるビジネス・オブジェクトのタイプと動詞は、WebSphere MQ メッセージ・ヘッダーに含まれる FORMAT フィールドによって決定されます。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属性を構成し、WebSphere MQ メッセージ・ヘッダーの FORMAT フィールドのテキストに関連付けます。メタオブジェクト属性には、メッセージ処理のガイドラインも含まれます。

入力キューからメッセージが検索されると、コネクタは、FORMAT テキスト・フィールドに関連付けられているビジネス・オブジェクト名を調べます。次に、ビジネス・オブジェクト名とともに、メッセージがデータ・ハンドラーに渡されます。ビジネス・オブジェクトにメッセージの内容が正常に取り込まれると、コネクタはそのビジネス・オブジェクトがサブスクライブされているかどうかをチェックしてから、`gotAppEvents()` メソッドを使用して統合ブローカーにデリバリーします。

コネクタは、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。

- 静的なコネクタ・メタオブジェクト
- 動的な子メタオブジェクト

動的な子メタオブジェクトの属性値は、静的なメタオブジェクトの属性値と重複し、それらをオーバーライドします。

ご自分の実装にはどちらのメタオブジェクトが最適であるかを判断する際には、以下のことを考慮してください。

• 静的メタオブジェクト

- 各種メッセージのメタデータがすべて固定されており、かつ構成時に指定可能である場合に役立ちます。
- ビジネス・オブジェクト・タイプごとにしか値を指定できません。例えば、Customer タイプのオブジェクトのすべてが同一の宛先に送られます。

• 動的メタオブジェクト

- ビジネス・プロセスからメッセージ・ヘッダー内の情報にアクセスできます。
- ビジネス・オブジェクト・タイプに関係なく、実行時にビジネス・プロセスでメッセージの処理を変更できます。例えば、動的メタオブジェクトを使用すると、アダプターに送信された Customer タイプのオブジェクトのそれぞれに、別々の宛先を指定することができます。
- サポートするビジネス・オブジェクトの構造を変更する必要があります。この変更によって、マップとビジネス・プロセスの変更が必要になる場合があります。
- カスタム・データ・ハンドラーに変更を加える必要があります。

メタオブジェクト・プロパティ

表 1 に、メタオブジェクトでサポートされるプロパティをすべて含むリストを示します。メタオブジェクトを実装するときには、これらのプロパティの説明を参照してください。メタオブジェクトには、表 1 に示すプロパティが 1 つ以上含まれていなければなりません。

一部のプロパティは、静的メタオブジェクトと動的メタオブジェクトのいずれかでしか使用できません。また、メッセージ・ヘッダーからの読み取りやメッセージ・ヘッダーへの書き込みが不可能なプロパティもあります。特定のプロパティ

ーについて、コネクタでどのように解釈および使用されるかを判断するには、1 ページの『第 1 章 概要』のイベント処理と要求処理に関する適切なセクションを参照してください。

表 1. WebSphere MQ アダプター・メタオブジェクトのプロパティ

プロパティ名	静的メタオブジェクトで定義可能か	動的メタオブジェクトで定義可能か	説明
CollaborationName	はい	いいえ	<p>CollaborationName は、ビジネス・オブジェクトと動詞の組み合わせに対する属性のアプリケーション固有のテキスト内で指定される必要があります。例えば、ユーザーが動詞 Create 付きのビジネス・オブジェクト Customer の同期イベントを処理しようとしている場合、静的メタデータ・オブジェクトは Customer_Create という名前の属性を含んでいる必要があります。</p> <p>Customer_Create 属性は、名前と値のペアを含むアプリケーション固有のテキストを含んでいる必要があります。例えば、CollaborationName=MyCustomerProcessingCollab です。構文の詳細については、39 ページの『静的メタオブジェクトの作成の概要』のセクションを参照してください。</p> <p>この条件が満たされていない場合は、コネクタが Customer ビジネス・オブジェクトに関する要求を同期的に処理しようとするランタイム・エラーが発生します。</p> <p>注: このプロパティは、同期要求にのみ利用可能です。</p>
DataHandlerConfigMO	はい	はい	<p>構成情報を提供するために、データ・ハンドラーに渡されるメタオブジェクト。静的なメタオブジェクトに指定された場合、この値は DataHandlerConfigMO コネクタ・プロパティに指定された値をオーバーライドします。このメタオブジェクト・プロパティは、さまざまなタイプのビジネス・オブジェクトを処理するために複数の異なるデータ・ハンドラーが必要な場合に使用します。データ形式が実際のビジネス・データに依存する可能性がある場合は、要求処理には動的な子メタオブジェクトを使用します。指定するビジネス・オブジェクトは、コネクタ・エージェントでサポートされるものでなければなりません。83 ページの『付録 B. Connector Configurator』の説明を参照してください。</p>

表 1. WebSphere MQ アダプター・メタオブジェクトのプロパティ (続き)

プロパティ名	静的メタオブジェクトで定義可能か	動的メタオブジェクトで定義可能か	説明
DataHandlerMimeType	はい	はい	使用すると、特定の MIME タイプに基づいたデータ・ハンドラーを要求できます。メタオブジェクトに指定されている場合、この値は DataHandlerMimeType コネクタ・プロパティに指定されている値をオーバーライドします。このメタオブジェクト・プロパティは、さまざまなタイプのビジネス・オブジェクトを処理するために複数の異なるデータ・ハンドラーが必要な場合に使用します。データ形式が実際のビジネス・データに依存する可能性がある場合は、要求処理には動的な子メタオブジェクトを使用します。 DataHandlerConfigMO に指定されたビジネス・オブジェクトは、このプロパティの値に対応する属性を含める必要があります。83 ページの『付録 B. Connector Configurator』の説明を参照してください。 83 ページの『付録 B. Connector Configurator』の説明を参照してください。
DataHandlerClassName	はい	はい	特定のビジネス・オブジェクトに関連付けるインバウンド (イベント) メッセージのフォーマットまたはタイプです。この値は、メッセージ内容の識別に役立つものであり、メッセージを生成するアプリケーションによって決まります。検索されたメッセージがこのフォーマットである場合、そのメッセージは (可能であれば) 特定のビジネス・オブジェクトに変換されます。ビジネス・オブジェクトにこのフォーマットが指定されていない場合、コネクタは特定のビジネス・オブジェクトのサブスクリプション・デリバリーを処理しません。このプロパティを設定するときは、デフォルトのメタオブジェクト変換プロパティを使用しないでください。デフォルトのメタオブジェクト変換プロパティの値は、着信メッセージをビジネス・オブジェクトに一致させるために使用されます。コネクタがメッセージのフォーマットを定義していると見なすフィールドは、コネクタ固有のプロパティ MessageFormatProperty を使用してユーザーが定義できます。
InputFormat	はい	はい	アウトバウンド・メッセージに取り込まれるフォーマットです。OutputFormat 名の長さは 16 文字を超えないようにする必要があります。16 文字を超えると、WebSphere MQ がエラーを生成する可能性があります。OutputFormat が指定されていない場合、使用可能であれば入力フォーマットが使用されます。
OutputFormat	はい	はい	

表 1. WebSphere MQ アダプター・メタオブジェクトのプロパティ (続き)

プロパティ名	静的メタオブジェクトで定義可能か	動的メタオブジェクトで定義可能か	説明
InputQueue	はい	はい	<p>コネクタが、新しい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。このプロパティは、着信メッセージとビジネス・オブジェクトを一致させる目的でのみ使用されます。このプロパティを設定するときは、デフォルトの変換プロパティを使用しないでください。デフォルトの変換プロパティの値は、着信メッセージをビジネス・オブジェクトに一致させるために使用されます。</p> <p>注: コネクタ固有のプロパティにある InputQueue プロパティは、アダプターがポーリングするキューを定義します。これは、アダプターがポーリングするキューを決定するのに使用する唯一のプロパティです。静的 MO では、InputQueue プロパティおよび InputFormat プロパティは、アダプターが指定されたメッセージを特定のビジネス・オブジェクトにマップする条件として使用できます。この機能を実装するには、コネクタ固有のプロパティを使用して複数の入力宛先を構成し、さらに、必要に応じて、着信メッセージの入力フォーマットを基に各入力宛先に別個のデータ・ハンドラーをマップします。詳細については、40 ページの『データ・ハンドラーの入力キューへのマッピングの概要』を参照してください。</p>
OutputQueue	はい	はい	<p>特定のビジネス・オブジェクトから派生したメッセージが送信されるキューです。</p>
ResponseTimeout	はい	はい	<p>同期要求処理で応答を待機するときにタイムアウトとするまでの待機時間を、ミリ秒単位で示します。このプロパティが未定義のままか、またはゼロよりも小さい値に設定されている場合、コネクタは応答を待機せず、SUCCESS を即時に戻します。</p>
TimeoutFatal	はい	はい	<p>同期要求処理で、応答の受信がないためコネクタからエラー・メッセージを戻す動作が起動されるときに使用されます。このプロパティの値が True の場合、コネクタは、ResponseTimeout に指定されている時間内に応答の受信がなければ、APPRESPONSETIMEOUT をブローカーに戻します。このプロパティが未定義の場合、または False に設定されている場合、コネクタは応答タイムアウトが発生すると要求を失敗させます。ただし、終了させることはありません。デフォルト値は False です。</p>

表 1. WebSphere MQ アダプター・メタオブジェクトのプロパティ (続き)

プロパティ名	静的メタオブジェクトで定義可能か	動的メタオブジェクトで定義可能か	説明
DataEncoding	はい	はい	DataEncoding は、メッセージの読み取りおよび書き込みに使用されるエンコードです。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を読み取ろうとします。動的メタオブジェクトで定義された DataEncoding は、静的メタオブジェクトで定義された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。例えば、Text:ISO8859_1、Text:UnicodeLittle、Text、または Binary になります。このプロパティは内部的に InputFormat プロパティに関連します。InputFormat ごとに 1 つの DataEncoding のみを指定します。
以下に示すのは、JMS メッセージ・ヘッダーのみにマップされるフィールドです。具体的な説明や値の解釈などの詳細情報については、JMS API の仕様を参照してください。フィールドによっては、JMS プロバイダー間で解釈が異なることがあります。ご使用の JMS プロバイダーの資料を参照して、解釈の違いがないかどうかを確認してください。			
ReplyToQueue		はい	要求に対する応答メッセージの送信先となるキューです。
Type		はい	メッセージのタイプです。通常はユーザーが定義できます (JMS プロバイダーによって異なります)。
MessageID		はい	メッセージの固有 ID です (特定の JMS プロバイダーでのみ有効)。
CorrelationID	はい	はい	応答メッセージで、その応答の開始理由にあたる要求メッセージの ID を示すために使用されます。
Delivery Mode	はい	はい	メッセージを MOM システム内で永続化するかどうかを指定します。許容値は次のとおりです。 1 = 非永続 2 = 永続 その他の値も使用できる場合があります (JMS プロバイダーによって異なります)。
Priority		はい	メッセージの優先順位を数値で表現したものです。許容値は 0 から 9 です (数値が大きいほど優先順位が高まります)。
Destination		はい	MOM システム内でのメッセージの現在の位置 (除去済みの場合は除去直前の位置) です。
Expiration		はい	メッセージの存続時間です。
Redelivered		はい	以前に JMS プロバイダーからクライアントに対してメッセージのデリバリーが試行されたが、受取の確認がなかった可能性が高いことを示すものです。
Timestamp		はい	メッセージが JMS プロバイダーに渡された時刻です。
UserID		はい	メッセージを送信したユーザーの ID です。
AppID		はい	メッセージを送信したアプリケーションの ID です。
DeliveryCount		はい	デリバリーを試行した回数です。
GroupID		はい	メッセージ・グループの ID です。
GroupSeq		はい	GroupID に指定されたメッセージ・グループにおける順位です。
JMSProperties		はい	43 ページの『JMS プロパティ』を参照してください。

静的メタオブジェクトの作成の概要

WebSphere MQ アダプター構成のメタオブジェクトは、さまざまなビジネス・オブジェクト用に定義された変換プロパティのリストで構成されます。静的メタオブジェクトのサンプルを参照するには、Business Object Designer Express を起動し、アダプターに付属しているサンプル

connectors\¥WebSphereMQ¥samples¥LegacyContact¥WebSphereMQ_MO_Config.xsd を開きます。

コネクターがサポートする静的メタオブジェクトは、常に 1 つだけです。静的メタオブジェクトを実装するには、そのメタオブジェクトの名前を、コネクター・プロパティ ConfigurationMetaObject に指定します。

静的メタオブジェクトは、各属性が、それぞれ 1 つのビジネス・オブジェクトと動詞の組み合わせを、そのオブジェクトの処理に関連するメタデータのすべてとともに示す構造になっています。各属性の名前は、Customer_Create のように、ビジネス・オブジェクト・タイプと動詞の間を下線で区切った名前にする必要があります。属性のアプリケーション固有情報には、このオブジェクトと動詞の固有の組み合わせに対して指定するメタデータ・プロパティを表す名前と値のペアを、セミコロンで区切って 1 つ以上含める必要があります。

表 2. 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
<business object type>_<verb>	property=value;property=value;...
<business object type>_<verb>	property=value;property=value;...

例えば、次のようなメタオブジェクトがあるとします。

表 3. 静的メタオブジェクト構造のサンプル

属性名	アプリケーション固有情報
Customer_Create	OutputFormat=CUST;OutputDestination=QueueA
Customer_Update	OutputFormat=CUST;OutputDestination=QueueB
Order_Create	OutputFormat=ORDER;OutputDestination=QueueC

このサンプルのメタオブジェクトは、タイプが Customer で動詞が Create の要求ビジネス・オブジェクトを受信したときは、そのオブジェクトをフォーマットが CUST のメッセージに変換して宛先 QueueA に入れる必要があることを、コネクターに対して通知します。Customer タイプのオブジェクトの動詞が Update である場合、変換後のメッセージは QueueB に入れられます。オブジェクトのタイプが Order で動詞が Create である場合には、コネクターはそのオブジェクトをフォーマットが ORDER のメッセージに変換し、QueueC にデリバリーします。それ以外のビジネス・オブジェクトがコネクターに渡された場合には、アンサブスクライブされているものとして処理されます。

また、Default 属性を 1 つ指定して、アプリケーション固有情報 (ASI) のプロパティを 1 つ以上割り当てることもできます。このデフォルト属性のプロパティと、オブジェクトと動詞の組み合わせごとの属性のプロパティが結合されて、メタオブジェクトの属性の最終的なプロパティになります。オブジェクトと動詞の

組み合わせに関係なく汎用的に適用するプロパティが 1 つ以上ある場合には、このデフォルト属性を使用すると便利です。次の例の場合、コネクタは、オブジェクトと動詞の組み合わせ Customer_Create と Order_Create については、それぞれに個別に指定されているメタオブジェクト・プロパティに加えて OutputDestination=QueueA が指定されていると見なします。

表 4. 静的メタオブジェクト構造のサンプル

属性名	アプリケーション固有情報
デフォルト	OutputDestination=QueueA
Customer_Update	OutputFormat=CUST
Order_Create	OutputFormat=ORDER

静的メタオブジェクトにアプリケーション固有情報として指定できるプロパティは、35 ページの表 1 に記載されています。

注: 静的なメタオブジェクトが指定されていない場合、コネクタはポーリング中にある特定のメッセージ・フォーマットを特定のビジネス・オブジェクト・タイプにマップできません。この場合、コネクタはビジネス・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のみに基づいたビジネス・オブジェクトを作成できない場合、コネクタはこのメッセージ・フォーマットが認識されていないことを表すエラーを報告します。

静的メタオブジェクトの作成のステップ

静的メタオブジェクトを実装するには、以下の手順を実行します。

1. Business Object Designer Express を起動します。詳細については、「ビジネス・オブジェクト開発ガイド」を参照してください。
2. サンプルのメタオブジェクト
connectors¥WebSphereMQ¥samples¥LegacyContact¥Sample_WebSphereMQ_MO_Config.xsd を開きます。
3. 必要な要件が反映されるように属性と ASI を編集して (35 ページの表 1 を参照)、メタオブジェクト・ファイルを保管します。
4. 保管したメタオブジェクト・ファイルの名前を、コネクタ・プロパティ ConfigurationMetaObject の値に指定します。

データ・ハンドラーの入力キューへのマッピングの概要

静的メタオブジェクトのアプリケーション固有情報で InputQueue プロパティを使用することにより、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、異なる書式や変換要件を持つ複数の取引先と取り引きする場合に役立ちます。

データ・ハンドラーの入力キューへのマッピングのステップ

データ・ハンドラーを InputQueue にマップするには、以下の手順を実行します。

1. コネクタ固有のプロパティ (28 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. **Business Object Designer Express** で、適切な静的メタオブジェクトを開きます。
3. 静的メタオブジェクトのそれぞれの入力キューについて、アプリケーション固有情報に、キュー・マネージャー、入力キュー名、データ・ハンドラーのクラス名、および MIME タイプを指定します。

例えば、次に示す静的メタオブジェクトの属性は、データ・ハンドラーと、CompReceipts という名前の InputQueue を関連付けています。

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

動的子メタオブジェクトの作成の概要

静的メタオブジェクトを使用して必要なメタデータを指定することが困難または実行不可能な場合には、実行時に、ビジネス・オブジェクト・インスタンスごとにデリバリーされるメタデータをコネクターで受信することもできます。

動的メタオブジェクトを使用すると、要求処理では、コネクターがビジネス・オブジェクトの処理に使用するメタデータを要求ごとに変更できるようになります。また、イベント処理では、イベント・メッセージに関する情報の検索が可能になります。

コネクターは、コネクター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た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。この動的な子メタオブジェクトの属性値は、コネクターの構成に使用される静的なメタオブジェクトに指定可能であった変換プロパティと重複します。

動的な子メタオブジェクトのプロパティは静的なメタオブジェクトから検出されるプロパティをオーバーライドするため、動的な子メタオブジェクトを指定する場合は、静的なメタオブジェクトを指定するコネクター・プロパティを組み込む必要はありません。したがって、動的な子メタオブジェクトは、静的なメタオブジェクトとは無関係に使用することができ、その逆もまた同様です。

動的メタオブジェクトにアプリケーション固有情報として指定できるプロパティは、35 ページの表 1 に記載されています。

以下の属性は JMS および WebSphere MQ ヘッダー・プロパティを反映しており、動的メタオブジェクトで認識されます。

表 5. 動的メタオブジェクト・ヘッダー属性

ヘッダー属性名	モード	対応する JMS ヘッダー
CorrelationID	読み取り/書き込み	JMSCorrelationID

表 5. 動的メタオブジェクト・ヘッダー属性 (続き)

ヘッダー属性名	モード	対応する JMS ヘッダー
ReplyToQueue	読み取り/書き込み	JMSReplyTo
DeliveryMode	読み取り/書き込み	JMSDeliveryMode
Priority	読み取り/書き込み	JMSPriority
Destination	読み取り	JMSDestination
Expiration	読み取り	JMSExpiration
MessageID	読み取り	JMSMessageID
Redelivered	読み取り	JMSRedelivered
TimeStamp	読み取り	JMSTimeStamp
Type	読み取り	JMSType
UserID	読み取り	JMSXUserID
AppID	読み取り	JMSXAppID
DeliveryCount	読み取り	JMSXDeliveryCount
GroupID	読み取り	JMSXGroupID
GroupSeq	読み取り	JMSXGroupSeq
JMSProperties	読み取り/書き込み	

読み取り専用属性は、イベント通知中にメッセージ・ヘッダーから読み取られ、動的メタオブジェクトに書き込まれます。これらのプロパティーは、要求処理中に応答メッセージが発行されたときに動的メタオブジェクトも設定します。読み取り/書き込み属性は、要求処理中に作成されるメッセージ・ヘッダーで設定されます。イベント通知中は、読み取り/書き込み属性はメッセージ・ヘッダーから読み取られ、動的メタオブジェクトを設定します。

動的メタオブジェクトは、各属性がそれぞれ 1 つのメタデータ・プロパティーと値を meta-object property name =meta-object property value の形式で表す構造になっています。

注: IBM WebSphere の標準のデータ・ハンドラーは、いずれも、cw_mo_ タグを認識すると、その後に指定されている動的メタオブジェクトを表す属性をビジネス・データ用の属性として処理しないように設計されています。アダプターで使用するカスタム・データ・ハンドラーを開発するときは、同様に設計する必要があります。

ポーリング時の動的子メタオブジェクトの取り込み

ポーリング中に検索されたメッセージについてさらに詳しい情報をコラボレーションに提供するため、コネクタは、作成されたビジネス・オブジェクトに動的なメタオブジェクトが定義済みである場合、その特定の属性に値を取り込みます。

表 6 に、動的子メタオブジェクトがポーリングのためにどのように構成されるかを示します。

表 6. ポーリング用の動的子メタオブジェクトの構造

プロパティー名	サンプル値
InputFormat	CUST_IN

表 6. ポーリング用の動的子メタオブジェクトの構造 (続き)

プロパティ名	サンプル値
InputQueue	MYInputQueue
OutputFormat	CxIgnore
OutputQueue	CxIgnore
ResponseTimeout	CxIgnore
TimeoutFatal	CxIgnore

表 6 に示すように、動的子メタオブジェクトに、追加属性 `Input_Format` および `InputQueue` を定義することができます。`Input_Format` には検索されたメッセージのフォーマットが取り込まれ、`InputQueue` 属性には特定のメッセージが検索されたキューの名前が含まれます。これらのプロパティが子メタオブジェクトで定義されていない場合は、これらのプロパティは取り込まれません。

シナリオ例:

- コネクターは、キュー `MyInputQueue` からフォーマット `CUST_IN` で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調べてメタオブジェクトが定義されているかどうかを判断します。
- 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、定義に基づいて `InputQueue` および `InputFormat` 属性に値を取り込んで、ビジネス・オブジェクトを使用可能なコラボレーションにパブリッシュします。

JMS ヘッダーと動的子メタオブジェクトの属性

動的メタオブジェクトに属性を追加すると、メッセージ・トランスポートの詳細情報を取得したりメッセージ・トランスポートを詳細に制御したりすることができます。このセクションでは、これらの属性、および属性がイベント通知と要求処理に及ぼす影響について説明します。

JMS プロパティ: 動的メタオブジェクトの他の属性と異なり、`JMSProperties` は単一カーディナリティ子オブジェクトを定義する必要があります。この子オブジェクトの各属性は、以下のように `JMS` メッセージ・ヘッダーの可変部分で読み取り/書き込みを行う単一プロパティを定義する必要があります。

1. 属性の名前はセマンティック値を持ちません。
2. 属性のタイプは、`JMS` プロパティ・タイプに無関係に必ず `String` でなければなりません。
3. 属性のアプリケーション固有情報は、属性をマップする `JMS` メッセージ・プロパティの名前と形式を定義する 2 つの名前と値の組を含まなければなりません。名前はユーザーが定義できます。値の型は次のいずれかでなければなりません。
 - `Boolean`
 - `String`
 - `Int`
 - `Float`

- Double
- Long
- Short
- Byte

以下の表に、JMSProperties オブジェクトの属性に対して定義する必要があるアプリケーション固有情報プロパティを示します。

表 7. JMS プロパティ属性のアプリケーション固有情報

属性	指定可能な値	ASI	コメント
名前	任意の有効な JMS プロパティ名 (有効とは、プロパティの型と ASI に定義した型が矛盾しないこと)	name=<JMS プロパティ名> ;type=<JMS プロパティの型>	ベンダーによっては、拡張機能を提供するために特定のプロパティを予約している場合があります。一般に、ユーザーはベンダー固有の機能にアクセスする場合以外は、JMS で開始するカスタム・プロパティを定義してはなりません。
Type	String	type=<コメントを参照>	これは JMS プロパティのタイプです。JMS API は、JMS メッセージに値を設定するための多くのメソッドを提供します (例: setIntProperty、setLongProperty、setStringProperty)。ここで指定する JMS プロパティのタイプによって、どのメソッドを使用してメッセージのプロパティ値を設定するかが決まります。

次の例では、Customer オブジェクトに JMSProperties 子オブジェクトを定義して、メッセージ・ヘッダーのユーザー定義フィールドにアクセスできるようにしています。

```
Customer (ASI = cw_mo_conn=MetaData)
  -- Id
  -- FirstName
  -- LastName
  -- ContactInfo
  -- MetaData
    -- OutputFormat = CUST
    -- OutputDestination = QueueA
    -- JMSProperties
      -- RoutingCode = 123 (ASI= name=RoutingCode;type=Int)
      -- Dept = FD (ASI= name=RoutingDept;type=String)
```

もう 1 つの例として、図 3 に、動的メタオブジェクトに含まれる属性 JMSProperties と、JMS メッセージ・ヘッダーの 4 つのプロパティ (ID、GID、RESPONSE、および RESPONSE_PERSIST) の定義を示します。属性のアプリケーション固有情報はそれぞれの名前およびタイプを定義します。例えば、属性 ID はタイプ String の JMS プロパティ ID にマップされます。

	Pos	Name	Type	Key	Reqd	Card	App Spec Info
1	1	JMSProperties	TeamCenter_JMS_Properties	<input type="checkbox"/>	<input type="checkbox"/>	1	
1.1	1.1	ID	String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		name=ID,type=String
1.2	1.2	GID	String	<input type="checkbox"/>	<input checked="" type="checkbox"/>		name=GID,type=String
1.3	1.3	RESPONSE	String	<input type="checkbox"/>	<input checked="" type="checkbox"/>		name=RESPONSE,type=Boolean
1.4	1.4	RESP_PERSIST	String	<input type="checkbox"/>	<input checked="" type="checkbox"/>		name=RESPONSE_PERSIST,type=Boolean
1.5	1.5	ObjectEventId	String	<input type="checkbox"/>	<input type="checkbox"/>		
2	2	OutputFormat	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

図 3. 動的メタオブジェクトの JMS プロパティ属性

動的メタオブジェクトの作成のステップ

動的メタオブジェクトを実装するには、以下の手順を実行します。

1. **Business Object Designer Express** を起動します。詳細については、「ビジネス・オブジェクト開発ガイド」を参照してください。
2. 処理するときに動的メタオブジェクトを作用させる必要があるトップレベル・ビジネス・オブジェクトを開きます。
3. 開いたトップレベル・オブジェクトに動的メタオブジェクトを子オブジェクトとして追加し、名前と値のペア `cw_mo_conn=<MO attribute>` をそのトップレベル・オブジェクトの ASI に追加します。ここで、`<MO attribute>` は、そのトップレベル・オブジェクトに含まれる、動的メタオブジェクトを表現する属性の名前です。以下に例を示します。

```
Customer (ASI = cw_mo_conn=MetaData)
  -- Id
  -- FirstName
  -- LastName
  -- ContactInfo
  -- MetaData
    -- OutputFormat = CUST
    -- OutputDestination = QueueA
```

コネクタは、上の定義にあてはまる要求を受け取ると、その要求 (Customer オブジェクト) をフォーマットが CUST のメッセージに変換し、キュー QueueA にそのメッセージを入れます。

4. トップレベル・ビジネス・オブジェクトを保管します。

注: 複数のビジネス・オブジェクトで同じ動的メタオブジェクトを使用することも、それぞれで異なる動的メタオブジェクトを使用することもできます。また、動的メタオブジェクトを一切使用しなくてもかまいません。

始動ファイルの構成

Windows および Linux の始動ファイルが構成され、ご使用の WebSphere MQ Java クライアント・ライブラリーが格納されているディレクトリーに設定されています。

始動

コネクターの始動

コネクターは、**コネクター始動スクリプト**を使用して明示的に開始する必要があります。Windows システムでは、始動スクリプトはコネクターのランタイム・ディレクトリー `ProductDir¥connectors¥connName` に存在していなければなりません。ここで、`connName` はコネクターを示します。

Linux システムでは、始動スクリプトは `ProductDir/bin` ディレクトリーに存在していなければなりません。

i5/OS システムでは、始動スクリプトは、コネクターの実行に使用する `/QIBM/UserData/WBIServer44/<instance>/connectors/<ConnInstance/` に存在しなければなりません。

表 8 が示すとおり、始動スクリプトの名前はオペレーティング・システム・プラットフォームにより異なります。

表 8. コネクターの始動スクリプト

オペレーティング・システム	始動スクリプト
Linux	connector_manager
i5/OS	start_connName.sh
Windows	start_connName.bat

始動スクリプトが実行されると、デフォルトで構成ファイルは `Productdir` にあることが要求されます (下記のコマンドを参照)。ここに構成ファイルを格納します。

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

• Windows システムでのコネクターの開始:

- 「スタート」メニューから、「プログラム」>「IBM WebSphere Business Integration Server Express」>「アダプター」>「コネクター」を選択します。デフォルトでは、プログラム名は「IBM WebSphere Business Integration Server Express」となっています。ただし、これはカスタマイズすることができます。または、コネクターへのデスクトップ・ショートカットを作成することもできます。
- Windows コマンド行から、次を入力します: `start_connName connName brokerName {-cconfigFile}`
- Windows システムでは、コネクターを Windows のサービスとして開始されるように構成できる。この場合、自動サービスでは Windows システムがブート

するとき、手動サービスでは「Windows サービス」ウィンドウからサービスを開始するときに、コネクターが開始されます。

- **Linux システムでのコネクターの開始:**

- コマンド行から、以下を入力します。

```
connector_manager -start connName brokerName [-cconfigFile ]
```

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

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

- **i5/OS システムでのコネクターの開始:**

- WebSphere Business Integrations Server Express Console がインストールされている Windows システムから、「**IBM WebSphere Business Integration Server Express**」 > 「**Toolset Express**」 >

「管理」 > 「コンソール」を選択します。次に、OS/400 または i5/OS システム名または IP アドレスと、*JOBCTL 特殊権限を持つユーザー・プロファイルおよびパスワードを指定します。コネクターのリストからコネクターを選択して、「開始」をクリックします。

- コンソールを使用してアダプターを自動的に開始するには、`submit_adapter.sh` スクリプトを使用します。これが、サーバーの自動開始ジョブ・エントリ内のサブシステムを使用してアダプターが開始する唯一の方法です。

- バッチ・モードでは、i5/OS コマンド行から CL コマンド QSH を実行し、QSHHELL 環境から `/QIBM/ProdData/WBIServer44/bin/submit_adapter.sh` *connName* *WebSphereICSName* *pathToConnNameStartScript* *jobDescriptionName* を実行する必要があります。ここで、*connName* はコネクター名、*WebSphereICSName* は Interchange Server Express サーバー名 (デフォルトは QWBIDFT44)、*pathToConnNameStartScript* はコネクター始動スクリプトの絶対パス、*jobDescriptionName* は QWBISVR44 ライブラリーで使用するジョブ記述の名前です。

- 対話モードでは、CL コマンド QSH を実行し、QSHHELL 環境から `/QIBM/UserData/WBIServer44/WebSphereICSName/connectors/connName/start_connName.sh` *connName* *WebSphereICSName* [-cConfigFile] を実行する必要があります。ここで、*connName* はコネクターの名前であり、*WebSphereICSName* は InterChange Server Express インスタンスの名前です。

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、「システム管理ガイド」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、コネクターが始動された方法によって異なります。

- **Windows:**

- コネクター用の別個の「コンソール」ウィンドウを作成する始動スクリプトを起動できます。このウィンドウで、「q」と入力して Enter キーを押すと、コネクターが停止します。

- コネクタを Windows のサービスとして始動するように構成できます。この場合、Windows システムがシャットダウンされるとコネクタは停止します。

- **i5/OS:**

- コンソールを使用して、または QSHELL で「submit_adapter.sh」スクリプトを使用してコネクタを始動した場合は、次の 2 つの方法のうちの 1 つを使用してコネクタを停止できます。
- WebSphere Business Integration Server Express Console がインストールされている Windows システムから、「**IBM WebSphere Business Integration Express**」>「**Toolset Express**」>「**管理**」>「**コンソール**」を選択します。次に、OS/400 または i5/OS システム名または IP アドレスと、*JOBCTL 特殊権限を持つユーザー・プロファイルおよびパスワードを指定します。リストから WebSphere MQ アダプターを選択して、「**停止**」ボタンを選択します。CL コマンド WRKACTJOB SBS (QWBISVR44) を使用して Server Express 製品に対するジョブを表示します。リストをスクロールして、コネクタのジョブ記述に一致するジョブ名を持つジョブを探します。例えば、WebSphere MQ コネクタの場合のジョブ名は QWBIWEBMQC です。このジョブに対してオプション 4 を選択し、F4 を押して ENDJOB コマンドのプロンプトを取得します。次に、オプション・パラメーターとして *IMMED を指定し、Enter を押します。

注: QWBISVR44 サブシステムが終了すると、コネクタは終了します。

- QSHELL から start_connName.sh スクリプトを使用してアダプターを始動した場合は、F3 を押してコネクタを終了します。
/QIBM/ProdData/WBIServer44/bin ディレクトリーにあるスクリプト *stop_adapter.sh* を使用して、エージェントを停止することもできます。

- **Linux:**

コネクタはバックグラウンドで実行されるので、個別のウィンドウはありません。その代わりに、以下のコマンドを実行してコネクタを停止します。

```
connector_manager -stop connName
```

ここで、*connName* はコネクタ名です。

第 3 章 ビジネス・オブジェクトの作成および変更

- 『アダプターのビジネス・オブジェクトの構造』
- 52 ページの『エラー処理』
- 53 ページの『トレース』

コネクターには、ビジネス・オブジェクトのサンプルのみが付属しています。システム・インテグレーター、コンサルタント、またはお客様が、ビジネス・オブジェクトを構築する必要があります。

コネクターは、メタデータ主導型コネクターです。WebSphere Business Integration システムのビジネス・オブジェクトにおいては、メタデータはアプリケーションに関するデータであり、ビジネス・オブジェクト定義に格納され、コネクターとアプリケーションの間の通信を支援します。メタデータ主導型コネクターは、コネクターにハードコーディングされている命令ではなく、ビジネス・オブジェクト定義にエンコードされているメタデータに基づいて、サポートする各ビジネス・オブジェクトを処理します。

ビジネス・オブジェクトのメタデータには、ビジネス・オブジェクトの構造、その属性プロパティの設定、およびアプリケーション固有情報の内容が含まれます。コネクターはメタデータ主導型のため、コネクターのコーディングを変更しなくても、新規ビジネス・オブジェクトや変更されたビジネス・オブジェクトを処理できます。しかし、コネクターに構成されたデータ・ハンドラーは、コネクターのビジネス・オブジェクトの構造、オブジェクトのカーディナリティー、アプリケーション固有情報のフォーマット、およびビジネス・オブジェクトのデータベース表記について、ある条件を前提として動作します。したがって、WebSphere MQ のビジネス・オブジェクトを作成または変更する場合、変更の内容はコネクターが従うべき規則に準拠している必要があります。準拠していない場合、コネクターは新規ビジネス・オブジェクトや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。

この章では、コネクターによるビジネス・オブジェクトの処理方法と、コネクターの前提事項について説明します。この情報は、新規ビジネス・オブジェクトをインプリメントする際のガイドとして利用できます。

アダプターのビジネス・オブジェクトの構造

アダプターをインストールした後で、ビジネス・オブジェクトを作成する必要があります。構成されるデータ・ハンドラーについての要件を除いては、ビジネス・オブジェクトの構造に関する要件はありません。コネクターが処理するビジネス・オブジェクトは、InterChange Server Express によって許可されている任意の名前を持つことができます。

アダプターはキューからメッセージを検索し、(メタオブジェクトによって定義されている) ビジネス・オブジェクトにメッセージの内容を取り込もうとします。厳密に言えば、コネクターはビジネス・オブジェクトの構造を制御したり、ビジネス・オブジェクトの構造に影響を及ぼすことはありません。それらの機能は、コネクタ

一のデータ・ハンドラーの要件と、メタオブジェクト定義によって提供されます。実際には、ビジネス・オブジェクト・レベルのアプリケーション情報はありませぬ。より正確に言えば、ビジネス・オブジェクトを検索して渡すときのコネクターの主な役割は、メッセージをビジネス・オブジェクトに変換する (およびその逆の) 処理中に発生するエラーをモニターすることです。

ビジネス・オブジェクト・プロパティのサンプル

このセクションでは、Name-Value データ・ハンドラーを持つコネクターのビジネス・オブジェクト・プロパティのサンプルを示します。

```
[ReposCopy]
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = Sample_WebSphereMQ_LegacyContact
Version = 1.0.0
```

```
[Attribute]
Name = ContactId
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = true
DefaultValue = 1001
IsRequiredServerBound = false
[End]
```

```
[Attribute]
Name = FirstName
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = Jim
IsRequiredServerBound = false
[End]
```

```
[Attribute]
Name = LastName
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = Smith
IsRequiredServerBound = false
[End]
```

```
[Attribute]
Name = OfficePhoneArea
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = 650
IsRequiredServerBound = false
[End]
```

```
[Attribute]
Name = OfficePhone
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
```

```

DefaultValue = 555-1234
IsRequiredServerBound = false
[End]
[Attribute]
Name = OfficePhoneExt
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = x100
IsRequiredServerBound = false
[End]
[Attribute]
Name = FaxArea
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = 650
IsRequiredServerBound = false
[End]
[Attribute]
Name = FaxPhone
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = 555-1235
IsRequiredServerBound = false
[End]
[Attribute]
Name = Department
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = Engineering
IsRequiredServerBound = false
[End]
[Attribute]
Name = Title
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = Software Engineer
IsRequiredServerBound = false
[End]
[Attribute]
Name = EmailAddr
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = jim.smith@crossworlds.com
IsRequiredServerBound = false
[End]
[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 0

```

```
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]
```

エラー処理

コネクタによって生成されたすべてのエラー・メッセージは、WebSphere MQConnector.txt という名前のメッセージ・ファイルに格納されます。(このファイルの名前は、コネクタ構成標準プロパティ `LogFileName` によって決定されます。) 各エラーはエラー番号が付けられ、その後にエラー・メッセージが表示されます。

```
Message number
Message text
```

コネクタは、以降のセクションで説明する方法で特定のエラーを処理します。

アプリケーション・タイムアウト

エラー・メッセージ「ABON_APPRESPONSETIMEOUT」は、以下の場合に戻されません。

- コネクタは、メッセージの検索中に JMS サービス・プロバイダーとの接続を確立できませんでした。
- コネクタはビジネス・オブジェクトをメッセージに正常に変換しましたが、接続切断が原因でメッセージを出力キューにデリバリーできませんでした。
- コネクタはメッセージを発行しましたが、変換プロパティ `TimeoutFatal` が `True` であるビジネス・オブジェクトに対する応答の待機中にタイムアウトが発生しました。
- コネクタは、戻りコードが `APP_RESPONSE_TIMEOUT` または `UNABLE_TO_LOGIN` の応答メッセージを受信しました。

アンサブスクライブされたビジネス・オブジェクト

アンサブスクライブされたビジネス・オブジェクトに関連付けられているメッセージを検索した場合、コネクタは `UnsubscribedQueue` プロパティで指定されたキューにメッセージをデリバリーします。

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破棄されます。

`gotAppEvent()` メソッドによって `NO_SUBSCRIPTION_FOUND` コードが戻されると、コネクタは `UnsubscribedQueue` プロパティで指定されたキューにメッセージを送信し、他のイベントの処理を続けます。

コネクタがアクティブでない

`gotAppEvent()` メソッドが `CONNECTOR_NOT_ACTIVE` コードを戻すと、`pollForEvents()` メソッドは `APP_RESPONSE_TIMEOUT` コードを戻し、イベントは `InProgress` キューに置かれたままになります。

データ・ハンドラーの変換

データ・ハンドラーがメッセージからビジネス・オブジェクトへの変換に失敗した場合、または、(JMS プロバイダーではなく) ビジネス・オブジェクトに固有の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デリバリーされます。`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破棄されます。

データ・ハンドラーがビジネス・オブジェクトからメッセージへの変換に失敗した場合、`BON_FAIL` が戻されます。

トレース

トレースはオプションのデバッグ機能で、オンにするとコネクタの動作を詳細にトレースできます。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・メッセージの構成の詳細については、21 ページの『第 2 章 アダプターのインストールおよび構成』のコネクタ構成プロパティの説明を参照してください。

コネクタのトレース・メッセージに推奨される内容を以下に示します。

- レベル 0 このレベルは、コネクタのバージョンを確認するトレース・メッセージに使用します。
- レベル 1 このレベルは、処理される各ビジネス・オブジェクトについての重要な情報を提供するトレース・メッセージや、ポーリング・スレッドが入力キュー内で新しいメッセージを検出するたびに記録されるトレース・メッセージに使用します。
- レベル 2 このレベルは、ビジネス・オブジェクトが `gotAppEvent()` または `executeCollaboration()` から `InterChange Server Express` に送付されるたびに記録されるトレース・メッセージに使用します。
- レベル 3 このレベルは、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するトレース・メッセージや、出力キューへのメッセージのデリバリーに関する情報を提供するトレース・メッセージに使用します。

- レベル 4 このレベルは、コネクタが動作を開始または終了した時間を記録するトレース・メッセージに使用します。
- レベル 5 このレベルは、コネクタの初期化を示すトレース・メッセージ、アプリケーション内で実行されるステートメントを表すトレース・メッセージ、メッセージが除去されるかキューに送出されるたびに記録されるトレース・メッセージ、または、ビジネス・オブジェクトのダンプを記録するトレース・メッセージに使用します。
- このレベルを使用して、アダプターによってキャッチされた例外の `printStackTrace()` をダンプします。

第 4 章 トラブルシューティング

この章では、コネク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

初期化中にコネク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...

初期化中にコネク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...

初期化中にコネク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...

初期化中にコネクターが予期しないエラーでシャットダウンし、次の例外が報告されました:
java.lang.UnsatisfiedLinkError: no mqjbnd01 in shared library path

コネクターから次の例外が報告されました: MQJMS2005: failed to create MQQueueManager for ':'

考えられる処置/説明

コネクターは、IBM WebSphere MQ Java クライアント・ライブラリーからファイル `.jms.jar` を見つけることができません。start_connector.bat 内の変数 `WebSphere MQ_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。

コネクターは、IBM WebSphere MQ Java クライアント・ライブラリーからファイル `com.ibm.mqjms.jar` を見つけることができません。start_connector.bat 内の変数 `WebSphere MQ_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。

コネクターは、IBM WebSphere MQ Java クライアント・ライブラリーからファイル `jndi.jar` を見つけることができません。start_connector.bat 内の変数 `WebSphere MQ_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。

コネクターは、IBM WebSphere MQ Java クライアント・ライブラリーから必要なランタイム・ライブラリー (`mqjbnd01.dll` [NT] または `libmqjbnd01.so` [Linux]) 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。

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

イベント処理

問題

コネクターは、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。

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、21 ページの『第 2 章 アダプターのインストールおよび構成』を参照してください。

問題

コネクタは、コネクタ・メタオブジェクト内でのメッセージ・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

これは WebSphere MQ MQMD メッセージ・ヘッダーの制限であり、コネクタ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 Server Express 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ています。説明は、InterChange Server Express が対象となります。

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

新規プロパティ

本リリースには、次の標準プロパティが追加されました。

- AdapterHelpName
- BiDi.Application
- BiDi.Broker
- BiDi.Metadata
- BiDi.Transformation
- ControllerEventSequencing
- jms.ListenerConcurrency
- jms.TransportOptimized
- TivoliTransactionMonitorPerformance

標準コネクタ・プロパティの概要

コネクタには 2 種類の構成プロパティがあります。

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

これらのプロパティは、アダプターのフレームワークおよびエージェントの実行時の振る舞いを決定します。

このセクションでは、Connector Configurator Express の始動方法について説明し、すべてのプロパティに共通する特性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ズ・ガイドを参照してください。

Connector Configurator Express の始動

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

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

Linux 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、Linux の統合ブローカーに接続してから、コネクタ用の Connector Configurator Express を開く必要があります。

構成プロパティ値の概要

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。

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

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。STRING プロパティ・タイプの長さに制限はありません。INTEGER タイプの長さは、アダプターを実行しているサーバーによって決まります。

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。

プロパティの更新特性 (すなわちコネクタ・プロパティへの変更を有効にする方法とタイミング) は、プロパティの性質によって異なります。

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、新しい値が即時に有効になります。ただし、コネクタがスタンドアロン・モードの場合 (System Manager に依存しない) です。
- **エージェント再始動 (InterChange Server Express のみ)**
コネクタ・エージェントを停止して再始動しなければ、新しい値が有効になりません。
- **コンポーネント再始動**
System Manager でコネクタを停止してから再始動しなければ、新しい値が有効になりません。エージェントまたはサーバー・プロセスを停止して再始動する必要はありません。
- **システム再始動**
コネクタ・エージェントおよびサーバーを停止して再始動しなければ、新しい値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 Express」ウィンドウ内の「更新メソッド」列を参照するか、59 ページの表 9 の「更新メソッド」列を参照してください。

標準プロパティが存在できる場所が 3 箇所あります。一部のプロパティは複数の場所にあってもかまいません。

- **ReposController**

このプロパティはコネクタ・コントローラ内にあり、その場所でのみ有効です。エージェント・サイドで値を変更した場合、コントローラには影響しません。

- **ReposAgent**

このプロパティはエージェント内にあり、その場所でのみ有効です。プロパティによっては、ローカル構成によってこの値をオーバーライドされることがあります。

- **LocalConfig**

このプロパティは、コネクタの構成ファイル内にあり、構成ファイルを通じてのみ機能することができます。コントローラはこのプロパティの値を変更することができず、システムが再配置されてコントローラが明示的に更新されなければ、構成ファイルに加えられた変更を認識しません。

標準プロパティの早見表

表9は、標準コネクタ構成プロパティの早見表です。すべてのコネクタでこれらのプロパティすべてを必要とするわけではなく、プロパティ設定は異なる場合があります。

各プロパティの説明については、表の次のセクションを参照してください。

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

表9. 標準構成プロパティの要約

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
AdapterHelpName	有効な <Regional Setting> ディレクトリーを含む <ProductDir>%bin%Data%App%Help 内の有効なサブディレクトリーのいずれか	テンプレート名 (有効な場合) またはブランク・フィールド	コンポーネント再始動	サポートされる地域設定。 chs_chn、cht_twn、 deu_deu、esn_esp、 fra_fra、ita_ita、 jpn_jpn、kor_kor、 ptb_bra、および enu_usa (デフォルト) を含む。
AdminInQueue	有効な JMS キュー名	<CONNECTORNAME> /ADMININQUEUE	コンポーネント再始動	このプロパティは、 DeliveryTransport の 値が JMS の場合のみ 有効です。
AdminOutQueue	有効な JMS キュー名	<CONNECTORNAME> /ADMINOUTQUEUE	コンポーネント再始動	このプロパティは、 DeliveryTransport の 値が JMS の場合のみ 有効です。
AgentConnections	1 から 4	1	コンポーネント再始動	このプロパティは、 DeliveryTransport の 値が MQ または IDL で、 RepositoryDirectory の値が <REMOTE> に設定され、 BrokerType の値が ICS の 場合のみ有効です。

表 9. 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
AgentTraceLevel	0 から 5	0	ICS では動的、 その他の場合は コンポーネント 再始動	
ApplicationName	アプリケーション名	コネクターの アプリケーション名に 指定された値	コンポーネント 再始動	
BiDi.Application	以下の双方向属性の 任意の有効な 組み合わせ 最初の文字: I、V 2 番目の文字: L、R 3 番目の文字: Y、N 4 番目の文字: S、N 5 番目の文字: H、C、N	ILYNN (5 文字)	コンポーネント 再始動	このプロパティは、 BiDi.Transformation の値が true の 場合のみ有効です。
BiDi.Broker	以下の双方向属性の 任意の有効な 組み合わせ 最初の文字: I、V 2 番目の文字: L、R 3 番目の文字: Y、N 4 番目の文字: S、N 5 番目の文字: H、C、N	ILYNN (5 文字)	コンポーネント 再始動	このプロパティは、 BiDi.Transformation の値が true の 場合のみ有効です。 BrokerType の値が ICS の場合、 プロパティは 読み取り専用です。
BiDi.Metadata	以下の双方向属性の 任意の有効な 組み合わせ 最初の文字: I、V 2 番目の文字: L、R 3 番目の文字: Y、N 4 番目の文字: S、N 5 番目の文字: H、C、N	ILYNN (5 文字)	コンポーネント 再始動	このプロパティは、 BiDi.Transformation の値が true の 場合のみ有効です。
BiDi.Transformation	true または false	false	コンポーネント 再始動	このプロパティは、 BrokerType の値が WAS でない場合のみ 有効です。
BrokerType	ICS	ICS	コンポーネント 再始動	
CharacterEncoding	サポートされる任意の コード。 次のリストはその一部です。 ascii7、ascii8、SJIS、 Cp949、GBK、Big5、 Cp297、Cp273、Cp280、 Cp284、Cp037、Cp437。	ascii7	コンポーネント 再始動	このプロパティは、 C++ コネクタでのみ 有効です。
CommonEventInfrastruc ture	true または false	false	コンポーネント 再始動	

表9. 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
CommonEventInfrastructureURL	URL ストリング。例えば、 corbaloc:iiop: host:2809。	デフォルト値はありません。	コンポーネント 再始動	このプロパティは、 CommonEvent Infrastructure の値が true の場合のみ 有効です。
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント 再始動	このプロパティは、 RepositoryDirectory の 値が <REMOTE> に 設定されて、 BrokerType の値が ICS の場合のみ有効です。
ContainerManagedEvents	ブランクまたは JMS	ブランク	コンポーネント 再始動	このプロパティは、 DeliveryTransport の 値が JMS の場合のみ 有効です。
ControllerEventSequencing	true または false	true	動的	このプロパティは、 RepositoryDirectory の値が <REMOTE> に 設定され、BrokerType の 値が ICS の場合のみ 有効です。
ControllerStoreAndForwardMode	true または false	true	動的	このプロパティは、 RepositoryDirectory の値が <REMOTE> に 設定され、BrokerType の 値が ICS の場合のみ 有効です。
ControllerTraceLevel	0 から 5	0	動的	このプロパティは、 RepositoryDirectory の 値が <REMOTE> に 設定されて、BrokerType の 値が ICS の場合のみ 有効です。
DeliveryQueue	任意の有効な JMS キュー名	<CONNECTORNAME> /DELIVERYQUEUE	コンポーネント 再始動	このプロパティは、 DeliveryTransport の 値が JMS の場合のみ 有効です。
DeliveryTransport	IDL または JMS	RepositoryDirectory の 値が <REMOTE> の 場合は IDL。 それ以外の場合は JMS。	コンポーネント 再始動	RepositoryDirectory の 値が <REMOTE> ではない場合、 このプロパティの 有効な値は JMS のみ です。
DuplicateEventElimination	true または false	false	コンポーネント 再始動	このプロパティは、 DeliveryTransport の 値が JMS の場合 のみ有効です。
EnableOidForFlowMonitoring	true または false	false	コンポーネント 再始動	このプロパティは、 BrokerType の値が ICS の場合のみ有効です。

表9. 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
FaultQueue	任意の有効なキュー名	<CONNECTORNAME> /FAULTQUEUE	コンポーネント 再始動	このプロパティは、 DeliveryTransport の 値が JMS の場合のみ 有効です。
jms.FactoryClassName	CxCommon.Messaging.jms .IBMMQSeriesFactory、 CxCommon.Messaging .jms.SonicMQFactory、 または 任意の Java クラス名	CxCommon.Messaging. jms.IBMMQSeriesFactory	コンポーネント 再始動	このプロパティは、 DeliveryTransport の 値が JMS の場合 のみ有効です。
jms.ListenerConcurrency	1 から 32767	1	コンポーネント 再始動	このプロパティは、 jms.TransportOptimized の 値が true の場合 のみ有効です。
jms.MessageBrokerName	jms.FactoryClassName の 値が IBM の場合は、 crossworlds.queue.manager を使用します。	crossworlds.queue. manager	コンポーネント 再始動	このプロパティは、 DeliveryTransport の値が JMS の場合のみ有効です。
jms.NumConcurrent Requests	正整数	10	コンポーネント 再始動	このプロパティは、 DeliveryTransport の値が JMS の場合のみ 有効です。
jms.Password	任意の有効なパスワード		コンポーネント 再始動	このプロパティは、 DeliveryTransport の 値が JMS の場合のみ 有効です。
jms.TransportOptimized	true または false	false	コンポーネント 再始動	このプロパティは、 DeliveryTransport の 値が JMS で、 BrokerType の値が ICS の 場合のみ有効です。
jms.UserName	任意の有効な名前		コンポーネント 再始動	このプロパティは、 DeliveryTransport の 値が JMS の場合のみ有効です。
JvmMaxHeapSize	ヒープ・サイズ (メガバイト 単位)	128m	コンポーネント 再始動	このプロパティは、 RepositoryDirectory の値が <REMOTE> に 設定され、BrokerType の 値が ICS の場合のみ 有効です。
JvmMaxNativeStackSize	スタックのサイズ (キロバイト 単位)	128k	コンポーネント 再始動	このプロパティは、 RepositoryDirectory の値が <REMOTE> に 設定され、BrokerType の 値が ICS の場合のみ 有効です。
JvmMinHeapSize	ヒープ・サイズ (メガバイト 単位)	1m	コンポーネント 再始動	このプロパティは、 RepositoryDirectory の値が <REMOTE> に 設定され、BrokerType の 値が ICS の場合のみ 有効です。

表9. 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
ListenerConcurrency	1 から 100	1	コンポーネント再始動	このプロパティは、DeliveryTransport の値が MQ の場合のみ有効です。
Locale	これは、サポートされるロケールの一部です。 en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR	en_US	コンポーネント再始動	
LogAtInterchangeEnd	true または false	false	コンポーネント再始動	このプロパティは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS の場合のみ有効です。
MaxEventCapacity	1 から 2147483647	2147483647	動的	このプロパティは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS の場合のみ有効です。
MessageFileName	有効なファイル名	InterchangeSystem.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	<CONNECTORNAME>/MONITORQUEUE	コンポーネント再始動	このプロパティは、DuplicateEventElimination の値が true で、ContainerManagedEvents に値がない場合にのみ有効です。
OADAutoRestartAgent	true または false	false	動的	このプロパティは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS の場合のみ有効です。
OADMaxNumRetry	正整数	1000	動的	このプロパティは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS の場合のみ有効です。
OADRetryTimeInterval	正整数 (単位: 分)	10	動的	このプロパティは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS の場合のみ有効です。
PollEndTime	HH = 0 から 23 MM = 0 から 59	HH:MM	コンポーネント再始動	

表9. 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
PollFrequency	正整数 (単位: ミリ秒)	10000	ブローカーが ICS の場合は動的。そうでない場合は、コンポーネント再始動。	
PollQuantity	1 から 500	1	エージェント再始動	このプロパティは、ContainerManagedEvents の値が JMS の場合のみ有効です。
PollStartTime	HH = 0 から 23 MM = 0 から 59	HH:MM	コンポーネント再始動	
RepositoryDirectory	ブローカーが ICS の場合は <REMOTE>。それ以外の場合は任意の有効なローカル・ディレクトリー。	ICS の場合、値は <REMOTE> に設定されます。	エージェント再始動	
RequestQueue	有効な JMS キュー名	<CONNECTORNAME>/REQUESTQUEUE	コンポーネント再始動	このプロパティは、DeliveryTransport の値が JMS の場合のみ有効です。
ResponseQueue	有効な JMS キュー名	<CONNECTORNAME>/RESPONSEQUEUE	コンポーネント再始動	このプロパティは、DeliveryTransport の値が JMS の場合のみ有効です。
RestartRetryCount	0 から 99	3	ICS の場合は動的、その他の場合はコンポーネント再始動	
RestartRetryInterval	1 から 2147483647 までの値 (分単位)。	1	ICS の場合は動的、その他の場合はコンポーネント再始動	
RHF2MessageDomain	mrm または xml	mrm	コンポーネント再始動	このプロパティは、DeliveryTransport の値が JMS で、WireFormat の値が CwXML の場合のみ有効です。
SourceQueue	任意の有効な WebSphere MQ キュー名	<CONNECTORNAME>/SOURCEQUEUE	エージェント再始動	このプロパティは、ContainerManagedEvents の値が JMS の場合のみ有効です。
SynchronousRequest Queue	任意の有効なキュー名	<CONNECTORNAME>/SYNCHRONOUSREQUEST QUEUE	コンポーネント再始動	このプロパティは、DeliveryTransport の値が JMS の場合のみ有効です。

表 9. 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
SynchronousRequest Timeout	0 から任意の数 (ミリ秒)	0	コンポーネント再始動	このプロパティは、DeliveryTransport の値が JMS の場合のみ有効です。
SynchronousResponse Queue	任意の有効なキュー名	<CONNECTORNAME>/SYNCHRONOUSRESPONSE QUEUE	コンポーネント再始動	このプロパティは、DeliveryTransport の値が JMS の場合のみ有効です。
TivoliMonitorTransaction Performance	true または false	false	コンポーネント再始動	
WireFormat	CwXML または CwBO	CwXML	エージェント再始動	RepositoryDirectory の値が <REMOTE> に設定されていない場合、このプロパティの値は、CwXML でなければなりません。RepositoryDirectory の値が <REMOTE> に設定されている場合、値は CwBO でなければなりません。
WsifSynchronousRequest Timeout	0 から任意の数 (ミリ秒)	0	コンポーネント再始動	BrokerType の値が ICS の場合、このプロパティは無効です。
XMLNameSpaceFormat	short または long	short	エージェント再始動	BrokerType の値が ICS の場合、このプロパティは無効です。

標準プロパティ

このセクションでは、標準コネクタ構成プロパティについて説明します。

AdapterHelpName

AdapterHelpName プロパティは、コネクタ固有の全般ヘルプ・ファイルがあるディレクトリの名前です。ディレクトリは、<ProductDir>¥bin¥Data¥App¥Help 内に配置される必要があり、少なくとも言語ディレクトリ enu_usa が含まれていなければなりません。ロケールに応じて、その他のディレクトリが含まれることがあります。

デフォルト値は、テンプレート名が有効であればテンプレート名、有効でなければブランクです。

AdminInQueue

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

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

AdminOutQueue

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

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

AgentConnections

AgentConnections プロパティは、ORB (オブジェクト・リクエスト・ブローカー) が初期化するときにかかれる ORB 接続の数を制御します。

このプロパティのデフォルト値は 1 です。

AgentTraceLevel

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

デフォルト値は 0 です。

ApplicationName

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

デフォルトはコネクタの名前です。

BiDi.Application

BiDi.Application プロパティは、このアダプターがサポートする任意のビジネス・オブジェクトの形式で、外部アプリケーションからアダプターに入ってくるデータの双方向フォーマットを指定します。このプロパティは、アプリケーション・データの双方向属性を定義します。これらの属性は以下のとおりです。

- テキストのタイプ: 暗黙または可視 (I または V)
- テキストの方向: 左から右または右から左 (L または R)
- 対称スワッピング: オンまたはオフ (Y または N)
- 成形 (アラビア語): オンまたはオフ (S または N)
- 数字成形 (アラビア語): ヒンディ語、コンテキスト、または名詞 (H、C、または N)

このプロパティは、**BiDi.Transformation** プロパティの値が `true` に設定されている場合のみ有効です。

デフォルト値は `ILYNN` (暗黙、左から右、オン、オフ、名詞) です。

BiDi.Broker

BiDi.Broker プロパティは、サポートされる任意のビジネス・オブジェクトの形式で、アダプターから統合ブローカーに送信されるデータの双方向フォーマットを指定します。データの双方向属性を定義します。属性は、前述の **BiDi.Application** の下にリストされています。

このプロパティは、**BiDi.Transformation** プロパティの値が **true** に設定されている場合のみ有効です。**BrokerType** プロパティが **ICS** の場合、プロパティ値は読み取り専用です。

デフォルト値は **ILYNN** (暗黙、左から右、オン、オフ、名詞) です。

BiDi.Metadata

BiDi.Metadata プロパティは、メタデータの双方向フォーマットまたは属性を定義します。メタデータは、外部アプリケーションへのリンクを確立および保守するために、コネクターが使用します。属性の設定は、双方向機能を使用する各アダプターに固有です。アダプターが双方向処理をサポートする場合、詳細についてはアダプター固有のプロパティに関するセクションを参照してください。

このプロパティは、**BiDi.Transformation** プロパティの値が **true** に設定されている場合のみ有効です。

デフォルト値は **ILYNN** (暗黙、左から右、オン、オフ、名詞) です。

BiDi.Transformation

BiDi.Transformation プロパティは、システムが実行時に双方向変換を実行するかどうかを定義します。

プロパティ値が **true** に設定されている場合、**BiDi.Application**、**BiDi.Broker**、および **BiDi.Metadata** プロパティが使用可能です。プロパティ値が **false** に設定されている場合は、それらは非表示になります。

デフォルト値は **false** です。

BrokerType

BrokerType プロパティは使用する統合ブローカー・タイプを指定します。値は、**ICS** (InterChange Server Express) です。

CharacterEncoding

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

注: Java ベースのコネクターでは、このプロパティは使用しません。C++ ベースのコネクターでは、このプロパティに **ascii7** という値を使用します。

デフォルトでは、サポートされる文字エンコードの一部のみが表示されます。サポートされる他の値をリストに追加するには、製品ディレクトリー (<*ProductDir*>) に

ある `¥Data¥Std¥StdConnProps.xml` ファイルを手動で変更する必要があります。詳細については、本書の付録『Connector Configurator Express』を参照してください。

ConcurrentEventTriggeredFlows

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

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

ソース・アプリケーションから宛先アプリケーションまでのフロー全体に並行処理を実装するには、以下のプロパティを構成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、複数のスレッドを使用できるようにコラボレーションを構成する必要があります。
- 宛先アプリケーションのアプリケーション固有コンポーネントを、複数の要求を並行して処理できるように構成する必要があります。

`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ーのポーリングでは無効です。

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

デフォルト値は 1 です。

ContainerManagedEvents

`ContainerManagedEvents` プロパティにより、JMS イベント・ストアを使用する JMS 対応コネクターが、保証付きイベント・デリバリーを提供できるようになります。保証付きイベント・デリバリーでは、イベントはソース・キューから除去され、1 つの JMS トランザクションとして宛先キューに配置されます。

このプロパティを 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設定する必要があります。

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

また、`MimeType` および `DHClass` (データ・ハンドラー・クラス) プロパティを設定したデータ・ハンドラーも構成する必要があります。`DataHandlerConfigMOName`

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

これらのプロパティはアダプター固有ですが、以下に値の例をいくつか示します。

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

「データ・ハンドラー」タブのこれらの値のフィールドは、ContainerManagedEvents プロパティを JMS という値に設定した場合にのみ表示されます。

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

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

デフォルト値はありません。

ControllerEventSequencing

ControllerEventSequencing プロパティは、コネクター・コントローラーでイベント順序付けを使用可能にします。

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

デフォルト値は `true` です。

ControllerStoreAndForwardMode

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

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

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

このプロパティを `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。

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

デフォルト値は `true` です。

ControllerTraceLevel

`ControllerTraceLevel` プロパティは、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を設定します。

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

デフォルト値は `0` です。

DeliveryQueue

`DeliveryQueue` プロパティは、コネクタが統合ブローカーへビジネス・オブジェクトを送信するときに使用するキューを定義します。

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

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

DeliveryTransport

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

- `RepositoryDirectory` プロパティの値が `<REMOTE>` に設定されている場合、`DeliveryTransport` プロパティの値には `IDL` または `JMS` を指定することができ、デフォルトは `IDL` です。
- `RepositoryDirectory` プロパティがローカル・ディレクトリーの場合、指定可能な値は `JMS` のみです。

`RepositoryDirectory` プロパティの値が `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。

デフォルト値は `JMS` です。

JMS

`JMS` トランスポート機構は、Java Messaging Service (`JMS`) を使用した、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すると、`jms.MessageBrokerName`、`jms.FactoryClassName`、`jms.Password`、`jms.UserName` などの追加の `JMS` プロパティ

イヤーが Connector Configurator Express にリストされます。jms.MessageBrokerName プロパティおよび jms.FactoryClassName プロパティは、このトランスポートの必須プロパティです。

InterChange Server Express (ICS) が統合ブローカーである場合、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、メモリー制限が発生することもあります。

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

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

このスクリプトは、製品ディレクトリー (<ProductDir>) の下の %bin ディレクトリーにあります。テキスト・エディターで、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 プロパティの値を 11 または 12 に設定する。このプロパティの詳細については、「*WebSphere Business Integration Server Express インストール・ガイド (Windows 版)*」、「*WebSphere Business Integration Server Express インストール・ガイド (Linux 版)*」、または「*WebSphere Business Integration Server Express インストール・ガイド (i5/OS 版)*」を参照してください。

DuplicateEventElimination

このプロパティの値が true の場合、JMS 対応コネクタでは重複イベントがデリバリー・キューヘデリバリーされないようにすることができます。この機能を使用するには、コネクタ開発時に、コネクタに対し、アプリケーション固有のコード内でビジネス・オブジェクトの ObjectEventId 属性として一意のイベント ID が設定されている必要があります。

注: このプロパティの値が true の場合、保証付きイベント・デリバリーを提供するには、MonitorQueue プロパティを使用可能にする必要があります。

デフォルト値は false です。

EnableOidForFlowMonitoring

このプロパティの値が true の場合、アダプター・ランタイムは、着信 ObjectEventID にフロー・モニターの外部キーのマークを付けます。

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

デフォルト値は `false` です。

FaultQueue

コネクターでメッセージを処理中にエラーが発生すると、コネクターは、そのメッセージ (および状況標識と問題説明) を `FaultQueue` プロパティで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。

jms.FactoryClassName

`jms.FactoryClassName` プロパティは、JMS プロバイダーのためにインスタンスを生成するクラス名を指定します。`DeliveryTransport` プロパティの値が JMS に設定されている場合、このプロパティを設定する必要があります。

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

jms.ListenerConcurrency

`jms.ListenerConcurrency` プロパティは、JMS コントローラーの並行リスナーの数を指定します。コントローラー内部で、並行してメッセージを取り出して処理するスレッドの数を指定します。

このプロパティは、`jms.OptimizedTransport` プロパティの値が `true` の場合のみ有効です。

デフォルト値は `1` です。

jms.MessageBrokerName

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

リモート・メッセージ・ブローカーに接続すると、このプロパティは次の値を要求します。

QueueMgrName:Channel:HostName:PortNumber

ここで、以下のように説明されます。

QueueMgrName はキュー・マネージャー名です。

Channel はクライアントが使用するチャンネルです。

HostName はキュー・マネージャーの配置先のマシン名です。

PortNumber はキュー・マネージャーが `listen` に使用するポートの番号です。

例えば、次のようにします。

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

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

jms.NumConcurrentRequests

`jms.NumConcurrentRequests` プロパティは、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はブロックされ、処理を続行するには他のいずれかの要求が完了するのを待機する必要があります。

デフォルト値は 10 です。

jms.Password

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

デフォルト値はありません。

jms.TransportOptimized

`jms.TransportOptimized` プロパティは、WIP (処理中の作業) が最適化されるかどうかを決定します。WIP を最適化するには、WebSphere MQ プロバイダーが必要です。最適化された WIP が作動するためには、メッセージング・プロバイダーが以下の操作を実行できなければなりません。

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

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

このプロパティは、`DeliveryTransport` の値が JMS で、`BrokerType` の値が ICS の場合のみ有効です。

デフォルト値は `false` です。

jms.UserName

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

デフォルト値はありません。

JvmMaxHeapSize

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

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

デフォルト値は 128M です。

JvmMaxNativeStackSize

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

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

デフォルト値は 128K です。

JvmMinHeapSize

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

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

デフォルト値は 1M です。

ListenerConcurrency

ListenerConcurrency 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 WebSphere MQ Listener でのマルチスレッド化をサポートしています。このプロパティにより、データベースへの複数イベントの書き込み操作をバッチ処理できるので、システム・パフォーマンスが向上します。

このプロパティは、MQ トランスポートを使用するコネクタのみで有効です。DeliveryTransport プロパティの値は MQ でなければなりません。

デフォルト値は 1 です。

Locale

Locale プロパティは、言語コード、国または地域、および、オプションに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、次の書式で指定します。

ll_TT.codeset

ここで、以下のように説明されます。

ll は、2 文字の言語コード (小文字を使用) です。

TT は、2 文字の国または地域コード (大文字を使用) です。

codeset は、関連文字コード・セットの名前です (オプションの場合があります)。

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

コネクタが国際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別するには、そのアダプターのユーザズ・ガイドを参照してください。

デフォルト値は en_US です。

LogAtInterchangeEnd

LogAtInterchangeEnd プロパティは、統合ブローカーのログ宛先にエラーを記録するかどうかを指定します。

ログ宛先にログを記録すると、E 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 ファイルで MESSAGE_RECIPIENT の値として指定された宛先に対する E メール・メッセージが生成されます。例えば、LogAtInterChangeEnd の値を 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、E メール・メッセージが送信されます。

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

デフォルト値は false です。

MaxEventCapacity

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

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

値は 1 から 2147483647 の間の正整数です。

デフォルト値は 2147483647 です。

MessageFileName

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

コネクタ・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: 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。

デフォルト値は InterchangeSystem.txt です。

MonitorQueue

MonitorQueue プロパティは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を指定します。

このプロパティは、DeliveryTransport プロパティの値が JMS で、DuplicateEventElimination の値が true の場合のみ有効です。

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

OADAutoRestartAgent

OADAutoRestartAgent プロパティはコネクタの使用する再始動機能が自動リモートかを指定します。この機能は、WebSphere MQ によりトリガーされる Object Activation Daemon (OAD) を使用して、異常シャットダウン後のコネクタの再始動や System Monitor からのリモート・コネクタの始動を行います。

自動およびリモートの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。WebSphere MQ によりトリガーされる OAD 機能の構成方法については、「*WebSphere Business Integration Server Express インストール・ガイド (Windows 版)*」、「*WebSphere Business Integration Server Express インストール・ガイド (Linux 版)*」、または「*WebSphere Business Integration Server Express インストール・ガイド (i5/OS 版)*」を参照してください。

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

デフォルト値は false です。

OADMaxNumRetry

OADMaxNumRetry プロパティは、異常シャットダウンの後で WebSphere MQ によりトリガーされる Object Activation Daemon (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。

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

デフォルト値は 1000 です。

OADRetryTimeInterval

OADRetryTimeInterval プロパティは、WebSphere MQ によりトリガーされる Object Activation Daemon (OAD) の再試行時間間隔の分数を指定します。コネクタ・エージェントがこの再試行間隔の間に再始動しないと、コネクタ・コントローラーが OAD にコネクタ・エージェントの再始動を再度要求します。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。

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

デフォルト値は 10 です。

PollEndTime

PollEndTime プロパティは、イベント・キューのポーリングを停止する時刻を指定します。形式は *HH:MM* です。ここで、*HH* は 0 から 23 時を表し、*MM* は 0 から 59 分を表します。

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

アダプター・ランタイムが以下のことを検出した場合、

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

PollFrequency プロパティに構成された値を使用してポーリングします。

PollFrequency

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

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

このプロパティでは、以下の値が有効です。

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

デフォルト値は 10000 です。

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

PollQuantity

PollQuantity プロパティは、コネクターがアプリケーションからポーリングする項目の数を指定します。アダプターにコネクター固有のポーリング数設定プロパティがある場合、標準プロパティの値は、このコネクター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。

このプロパティは、DeliveryTransport プロパティの値が JMS で、ContainerManagedEvents プロパティに値がある場合のみ有効です。

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

- 一度ポーリングされると、コネクタはメッセージの本文を検出し、それを添付ファイルとして読み取ります。本文の MIME タイプにはデータ・ハンドラーが指定されていないので、コネクタはメッセージを無視します。
- コネクタは最初の BO 添付ファイルを処理します。この MIME タイプには対応するデータ・ハンドラーがあるので、コネクタはビジネス・オブジェクトを Visual Test Connector に送信します。
- 二度目にポーリングされると、コネクタは 2 番目の BO 添付ファイルを処理します。この MIME タイプには対応するデータ・ハンドラーがあるので、コネクタはビジネス・オブジェクトを Visual Test Connector に送信します。
- それが受け入れられると、3 番目の BO 添付ファイルが送信されます。

PollStartTime

PollStartTime プロパティは、イベント・キューのポーリングを開始する時刻を指定します。形式は HH:MM です。ここで、HH は 0 から 23 時を表し、MM は 0 から 59 分を表します。

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

アダプター・ランタイムが以下のことを検出した場合、

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

PollFrequency プロパティに構成された値を使用してポーリングします。

RepositoryDirectory

RepositoryDirectory プロパティは、コネクタが 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保管されています。

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、コネクタが InterChange Server Express 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 の場合は、この値はデフォルトで <ProductDir>¥repository に設定されます。ただし、これには任意の有効なディレクトリー名を設定することができます。

RequestQueue

RequestQueue プロパティは、統合ブローカーがコネクタへビジネス・オブジェクトを送信するときに使用するキューを指定します。

このプロパティは、DeliveryTransport プロパティの値が JMS の場合のみ有効です。

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

ResponseQueue

ResponseQueue プロパティは、JMS 応答キューを指定します。JMS 応答キューは、応答メッセージをコネクタ・フレームワークから統合ブローカーヘデリバリーします。統合ブローカーが InterChange Server Express (ICS) の場合、サーバーは要求を送信し、JMS 応答キューの応答メッセージを待ちます。

このプロパティは、DeliveryTransport プロパティの値が JMS の場合のみ有効です。

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

RestartRetryCount

RestartRetryCount プロパティは、コネクタによるコネクタ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に接続されたコネクタに対して使用する場合、コネクタのマスター側のアプリケーション固有のコンポーネントがクライアント側のアプリケーション固有のコンポーネントの再始動を試行する回数が指定されます。

デフォルト値は 3 です。

RestartRetryInterval

RestartRetryInterval プロパティは、コネクタによるコネクタ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にリンクされたコネクタに対して使用する場合、コネクタのマスター側のアプリケーション固有のコンポーネントがクライアント側のアプリケーション固有のコンポーネントの再始動を試行する間隔が指定されます。

プロパティに使用可能な値の範囲は 1 から 2147483647 です。

デフォルト値は 1 です。

RHF2MessageDomain

RHF2MessageDomain 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 WebSphere Message Broker に送信するときに、アダプター・フレームワークにより JMS ヘッダー情報、ドメイン名、および固定値 mrm が書き込まれます。構成可能ドメイン名によって、WebSphere Message Broker がメッセージ・データを処理する方法を追跡できます。

ヘッダーの例を示します。

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

BrokerType の値が ICS の場合、このプロパティは無効です。また、このプロパティは、DeliveryTransport プロパティの値が JMS で、WireFormat プロパティの値が CwXML の場合のみ有効です。

可能な値は、mrm および xml です。デフォルト値は mrm です。

SourceQueue

SourceQueue プロパティは、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・フレームワーク用に、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。

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

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

SynchronousRequestQueue

SynchronousRequestQueue プロパティは、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デリバリーします。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、同期要求キューにメッセージを送信し、同期応答キューでブローカーからの応答を待機します。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。

このプロパティは、DeliveryTransport の値が JMS の場合のみ有効です。

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

SynchronousRequestTimeout

SynchronousRequestTimeout プロパティは、コネクタが同期要求への応答を待機する時間をミリ秒単位で指定します。コネクタは、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 (およびエラー・メッセージ) を障害キューに移動します。

このプロパティは、DeliveryTransport の値が JMS の場合のみ有効です。

デフォルト値は 0 です。

SynchronousResponseQueue

SynchronousResponseQueue プロパティは、同期要求に対する応答メッセージを、ブローカーからコネクタ・フレームワークにデリバリーします。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。

このプロパティは、DeliveryTransport の値が JMS の場合のみ有効です。

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

TivoliMonitorTransactionPerformance

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

デフォルト値は false です。

WireFormat

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

- RepositoryDirectory プロパティの値がローカル・ディレクトリーの場合、値は CwXML です。
- RepositoryDirectory プロパティの値がリモート・ディレクトリーの場合、値は CwB0 です。

付録 B. Connector Configurator

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

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

- コネクタ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
- 構成ファイルのプロパティを設定する

この付録では、次のトピックについて説明します。

- 『Connector Configurator Express の概要』
- 85 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』
- 88 ページの『新しい構成ファイルを作成』
- 92 ページの『構成ファイル・プロパティの設定』

Connector Configurator Express の概要

Connector Configurator Express によって、InterChange Server Express 統合ブローカーで使用するアダプターのコネクタ・コンポーネントを構成することができます。

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

- コネクタ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 1 つ構成ファイルを作成する必要があります。
- 構成ファイル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、InterChange Server Express の場合はコラボレーションとともに使用するマップを指定し、必要に応じてメッセージング、ロギング、トレース、およびデータ・ハンドラー・パラメーターを指定する必要があります。

コネクタ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に必要なプロパティ) 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、標準プロパティを最初から定義する必要はありません。ファイルを作成すると、Connector Configurator Express により標準プロパティがこの構成ファイルに挿入されます。ただし、Connector Configurator Express で各標準プロパティ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 Express 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、85 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。

Linux でのコネクタの実行

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

Connector Configurator Express 内のいくつかのプロパティはディレクトリー・パスを使用します。このパスは、Windows のディレクトリー・パスの規則がデフォルトになっています。Linux 環境で構成ファイルを使用する場合、これらのパスの Linux の規則に対応するように、ディレクトリー・パスを修正する必要があります。正しいオペレーティング・システム規則が拡張検証に使用されるように、ツールバー・ドロップ・リストでターゲット・オペレーティング・システムを選択します。

Connector Configurator Express の始動

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

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

スタンドアロン・モードでの Configurator の実行

どのブローカーを実行している場合にも、System Manager を実行せずに Connector Configurator Express を実行し、コネクタ構成ファイルを編集できます。

これを行うには、以下のステップを実行します。

- 「スタート」>「すべてのプログラム」から、「**IBM WebSphere Business Integration Express**」>「**Toolset Express**」>「開発」>「**Connector Configurator Express**」をクリックします。
- 「ファイル」>「新規」>「コネクタ構成」を選択します。

Connector Configurator Express を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存する方法が便利です (91 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

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

コネクタ固有のプロパティ・テンプレートの作成

コネクタの構成ファイルを作成するには、コネクタ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、または既存のコネクタ定義をテンプレートとして使用します。

- テンプレートの新規作成については、85 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。既存のテンプレートは `¥ProductDir¥bin¥Data¥App` ディレクトリーにあります。

新規テンプレートの作成

このセクションでは、テンプレートでプロパティを作成し、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。

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

1. 「ファイル」>「新規」>「コネクタ固有プロパティ・テンプレート (Connector-Specific Property Template)」とクリックします。
2. 「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。

- 「新規テンプレート名を入力してください」の下の「名前」フィールドに、新規テンプレートの名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
 - テンプレートに含まれているコネクタ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。
3. テンプレートを作成するときには、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。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。
 - 既存のテンプレートを変更する場合には、「変更する既存のテンプレートを選択してください: 検索テンプレート」の下の「テンプレート名」テーブルのリストから、テンプレート名を選択します。
 - このテーブルには、現在使用可能なすべてのテンプレートの名前が表示されます。テンプレートを検索することもできます。

一般特性の指定

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

「プロパティ・タイプ」がストリングの場合、「プロパティ・サブタイプ」を選択できます。これは、構成ファイルの保管時に構文検査を提供するオプションの値です。デフォルトは空白・スペースで、プロパティのサブタイプが指定されていないことを意味します。

プロパティ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。また、編集可能な値も設定できます。これを行うには、以下のステップを実行します。

1. 「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. 「プロパティを編集」表示でプロパティの名前を選択します。

3. 「最大長」および「最大複数値」のフィールドに値を入力します。

新規プロパティ値を作成するには、以下のステップを実行します。

1. 「値」列見出しの左側の正方形を右マウス・ボタンでクリックします。
2. ポップアップ・メニューから「追加」を選択して、「プロパティ値」ダイアログ・ボックスを表示します。ダイアログ・ボックスでは、プロパティ・タイプに応じて、値を入力するか、または値と範囲の両方を入力することができます。
3. 新規プロパティ値を入力し、「OK」をクリックします。右側の「値」パネルに値が表示されます。

「値」パネルには、3つの列からなるテーブルが表示されます。

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

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

「値の範囲」の列には、「プロパティ値」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・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・テンプレート」ダイアログ・ボックスが表示されます。

依存プロパティは、別のプロパティ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。プロパティを依存プロパティとして指定し、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」表示で、依存プロパティとして指定するプロパティを選択します。
2. 「プロパティを選択」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」表示で強調表示された依存プロパティで、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 (Finish)」をクリックします。入力した情報が、Connector Configurator Express によって、Connector Configurator Express がインストールされている %bin ディレクトリーの %data%app の下に XML 文書として保管されます。

パス名の設定

パス名の設定の一般的な規則のいくつかを以下に示します。

- Windows および Linux でのファイル名の最大長は 255 文字です。
- Windows では、絶対パス名は [Drive:][Directory]%filename の形式に従う必要があります。例えば、C:%WebSphereAdapters%bin%Data%Std%StdConnProps.xml のようにします。
Linux では、最初の文字は / でなければなりません。
- キュー名では、先頭または途中でスペースを使用することはできません。

新しい構成ファイルを作成

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

ファイルの拡張検証のために、オペレーティング・システムも選択します。ツールバーには「ターゲット・システム」というドロップ・リストがあり、ここで、プロパティの拡張検証用のターゲット・オペレーティング・システムを選択できます。選択可能なオプションは、「Windows」、「Linux」、および「i5/OS」、「その他」(Windows でも Linux でもない場合)、および「なし (拡張検証なし)」(拡張検証をオフに切り替え) です。始動時のデフォルトは「Windows」です。

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

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

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

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

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

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

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

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、そのテンプレートを使用して構成ファイルを作成できます。

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

• 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と一貫性をもつ一意の名前である必要があります。

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

• システム接続

「ICS」をクリックします。

• コネクタ固有プロパティ・テンプレートを選択 (Select Connector-Specific Property Template)

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。

4. 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。
5. ファイルを保管するには、「**ファイル**」>「**保管**」>「**ファイルに**」をクリックするか、「**ファイル**」>「**保管**」>「**プロジェクトに**」をクリックします。プロジェクトに保管するには、System Manager が実行中である必要があります。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 Express のメッセージ・パネルの状況表示に、構成ファイルが正常に作成されたことが示されます。

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

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

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

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

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

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

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

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

ICS 環境でのコネクター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ー定義が含まれている場合は、ファイルを開くとすべての定義が表示されません。

- 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 Express を始動します。
3. 「ファイル」>「開く」>「プロジェクトから」とクリックします。

構成ファイルの完成

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

構成画面のタイトルには、ファイル内で指定された統合ブローカーとコネクタ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 を選択します。
2. 選択したブローカーに関連付けられているコネクタ・プロパティが「標準のプロパティ」タブに表示されます。表に、「プロパティ名」、「値」、「タイプ」、「サブタイプ」（「タイプ」がストリングである場合）、「説明」、および「更新メソッド」が表示されます。
3. ここでファイルを保管するか、または 95 ページの『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。
4. 構成が完了したら、「ファイル」>「保管」>「プロジェクトに」を選択するか、または「ファイル」>「保管」>「ファイルに」を選択します。

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

複数のコネクタ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。

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

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

「ターゲット・システム」ドロップ・リストから「Windows」、「Linux」、および「i5/OS」、または「その他」を選択することによって拡張検証機能を使用する場合、システムはタイプだけでなくプロパティー・サブタイプを検証し、検証に失敗した場合は警告メッセージを表示します。

構成ファイル・プロパティーの設定

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

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

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

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

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

- 関連付けられたマップ
- セキュリティー

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

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

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

す。各コネクターには、そのコネクターのアプリケーションだけで使用されるアプリケーション固有のプロパティがあります。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。

「標準のプロパティ」と「コネクター固有プロパティ (Connector-Specific Properties)」のフィールドは、どのフィールドが構成可能であること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ーの特定のニーズによって異なります。値の変更も、これらのプロパティの除去も可能です。
- 「値」フィールドは構成できます。
- プロパティごとに「更新メソッド」フィールドが表示されます。これは、変更された値をアクティブにするためにコンポーネントまたはエージェントの再始動が必要かどうかを示します。この設定を構成することはできません。

標準コネクター・プロパティの設定

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る場合にはメニューから値を選択します。

注: プロパティの「タイプ」が「string」である場合、「サブタイプ」列にサブタイプ値が含まれている場合があります。このサブタイプは、プロパティの拡張検証に使用されます。

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

特定の標準プロパティに関する詳細を参照するには、「標準のプロパティ」タブ付きシート内のそのプロパティの「説明」列内の項目を左マウス・ボタンでクリックします。全般ヘルプをインストール済みの場合は、右側に矢印ボタンが表示されます。ボタンをクリックすると、「ヘルプ」ウィンドウが開き、標準プロパティの詳細が表示されます。

注: ホット・ボタンが表示されない場合、そのプロパティについては全般ヘルプが見つかりません。

インストール済みの場合、全般ヘルプ・ファイルは
<ProductDir>%bin%Data%Std%Help%<RegionalSetting>% にあります。

コネクタ固有の構成プロパティの設定

コネクタ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・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。「追加」をクリックしてプロパティを追加します。子プロパティを追加するには、親行番号を右マウス・ボタン・クリックして、「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。

注: プロパティの「タイプ」が「string」である場合、「サブタイプ」ドロップ・リストからサブタイプを選択できます。このサブタイプは、プロパティの拡張検証に使用されます。

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. 特定のプロパティに関する詳細を参照するには、そのプロパティの「説明」列内の項目を左マウス・ボタンでクリックします。全般ヘルプをインストール済みの場合は、ホット・ボタンが表示されます。ホット・ボタンをクリックすると、「ヘルプ」ウィンドウが開き、標準プロパティの詳細が表示されます。

注: ホット・ボタンが表示されない場合、そのプロパティについては全般ヘルプが見つかりません。

5. 93 ページの『標準コネクタ・プロパティの設定』で説明したように、変更内容を保管するかまたは破棄するかを選択します。

全般ヘルプ・ファイルがインストール済みで、AdapterHelpName プロパティがブランクである場合、Connector Configurator Express は、<ProductDir>%bin%Data%App%Help%<RegionalSetting>% にあるアダプター固有の全般ヘルプ・ファイルを指します。それ以外の場合、Connector Configurator Express は、<ProductDir>%bin%Data%App%Help%<AdapterHelpName>%<RegionalSetting>% にあるアダプター固有の全般ヘルプ・ファイルを指します。標準プロパティについての付録で説明されている AdapterHelpName プロパティを参照してください。

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

コネクター・プロパティの暗号化

「コネクター固有プロパティ」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、各コネクターのアダプター・ユーザー・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、すべての複数值が暗号化解除されます。

更新メソッド

57 ページの『標準コネクター・プロパティの概要』の下の標準プロパティの付録にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

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

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

ご使用のブローカーが InterChange Server Express の場合

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

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

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

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

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

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

エージェント・サポート: ビジネス・オブジェクトにエージェント・サポートがある場合、システムは、コネクタ・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・エージェントによってサポートされるよう指定するには、「エージェント・サポート」ボックスにチェックマークを付けます。「Connector Configurator Express」ウィンドウでは、「エージェント・サポート」を選択しても問題ないかどうかの検証は行われません。

最大トランザクション・レベル: コネクタの最大トランザクション・レベルは、そのコネクタ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の場合、選択可能な項目は「最大限の努力」のみです。

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

関連付けられたマップ

各コネクタは、ビジネス・オブジェクト定義とそれらに関連付けられたマップのうち現在 InterChange Server Express でアクティブであるものを示すリストをサポートします。このリストは、「関連付けられたマップ」タブを選択すると表示されます。

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

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

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

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的バインディング**

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

明示的バインディングが必要なのは、特定のサポートされるビジネス・オブジェクトに複数のマップが存在する場合のみです。InterChange Server Express は、ブート時、各コネクタのサポートされるビジネス・オブジェクトのそれぞれ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ています。

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

以下の手順を実行して、マップを明示的にバインドします。

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

セキュリティ

Connector Configurator Express 内の「セキュリティ」タブを使用して、メッセージにさまざまなプライバシー・レベルを設定することができます。DeliveryTransport プロパティが JMS に設定されている場合のみ、この機能を使用できます。

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

「鍵ストア・ターゲット・システムの絶対パス名」は、以下の値です。

- Windows の場合:
`<ProductDir>%connectors%security%<connectorname>.jks`
- Linux および i5/OS の場合:
`/ProductDir/connectors/security/<connectorname>.jks`

このパスおよびファイルは、コネクタを始動するシステム、すなわちターゲット・システム上に存在していなければなりません。

ターゲット・システムが現在実行中のシステムである場合のみ、右側の「参照」ボタンを使用できます。「プライバシー」が使用可能であり、メニュー・バーの「ターゲット・システム」が Windows に設定されている場合を除き、これはグレーアウトされています。

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

- “”: がデフォルトです。メッセージ・カテゴリにプライバシー・レベルが設定されていない場合に使用します。
- none。デフォルトと同じではありません。メッセージ・カテゴリにプライバシー・レベルなしと故意に設定する場合にこれを使用します。
- integrity
- privacy
- integrity_plus_privacy

「鍵の保守」機能によって、サーバーおよびアダプターの公開鍵を生成、インポート、およびエクスポートすることができます。

- 「鍵の生成」を選択すると、鍵を生成する keytool のデフォルトを含む「鍵の生成」ダイアログ・ボックスが表示されます。
- 「セキュリティ」タブの「鍵ストア・ターゲット・システムの絶対パス名」で入力した値が、鍵ストア値のデフォルトになります。
- 「OK」を選択すると、記入項目が検証され、鍵証明書が生成され、「Connector Configurator Express」ログ・ウィンドウに出力が送られます。

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

- エクスポート証明書のデフォルトは、ファイル拡張子が <filename>.cer であることを除き、鍵ストアと同じ値です。

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

- インポート証明書のデフォルトは、<ProductDir>%bin%ics.cer になります (システムにファイルが存在する場合)。
- インポート証明書関連はサーバー名でなければなりません。サーバーが登録されていれば、ドロップ・リストからそれを選択することができます。

DeliveryTransport の値が IDL の場合のみ、「アダプター・アクセス制御」機能が使用可能です。デフォルトでは、アダプターはゲスト ID を使用してログインします。「ゲスト ID の使用」ボックスにチェック・マークが付けられていない場合は、「アダプター ID」および「アダプター・パスワード」フィールドが使用可能です。

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

ログとトレースの値を変更するには、以下の手順を実行します。

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

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・メッセージを指定したファイルに書き込みます。ファイルを指定するには、ディレクトリー・ボタン (省略符号) をクリックし、指定する格納場所に移動し、ファイル名を指定し、「保

管」をクリックします。ログ・メッセージまたはトレース・メッセージは、指定した場所の指定したファイルに書き込まれます。

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

データ・ハンドラー

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

これらのプロパティーに使用する値については、付録 A の『コネクターの標準構成プロパティー』の `ContainerManagedEvents` の下の説明を参照してください。

構成ファイルの保管

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

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

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

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

構成ファイルの変更

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

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

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

- Connector Configurator Express で既存の構成ファイルを開きます。
- 「標準のプロパティ」タブを選択します。
- 「標準のプロパティ」タブの「ブローカー・タイプ」フィールドで、ご使用のブローカーに合った値を選択します。現行値を変更すると、プロパティ・ウィンドウ内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、コネクタの始動時にコネクタ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、コネクタが使用する始動ファイルを開き、コネクタ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 Express の使用

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

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

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

「CharacterEncoding」および「ロケール」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「ロケール」プロパティの値のリストにロケール `en_GB` を追加するには、`stdConnProps.xml` ファイルを開き、以下に太字で示される行を追加してください。

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

```
<Value>es_ES</Value>
<Value>pt_BR</Value>
<Value>en_US</Value>
<Value>en_GB</Value>
  <DefaultValue>en_US</DefaultValue>
</ValidValues>
</Property>
```

付録 C. チュートリアル

- 『チュートリアルについて』
- 104 ページの『始める前に』
- 104 ページの『環境のセットアップ』
- 107 ページの『シナリオの実行』
- 107 ページの『静的メタオブジェクトを使用したシナリオ』
- 108 ページの『動的メタオブジェクトを使用したシナリオ』

この付録では、WebSphere MQ を介して通信するアプリケーションとの間で、アダプターを使用してビジネス・オブジェクトを送受信する方法について説明します。このチュートリアルのシナリオは、アダプターの基本的な機能について説明することを目的としています。

表記規則のガイドについては、この文書のまえがきを参照してください。

注: MQ メッセージの格納および送信に使用できる、ダウンロード可能な独立した GUI ユーティリティの詳細については、<http://www.ibm.com/support> のサイトで「IH03: WBI Message Broker V5 - Message display, test and performance utilities」を検索してください。

チュートリアルについて

このチュートリアルは 2 つのシナリオから構成されています。1 つは静的メタオブジェクトを使用したシナリオで、もう 1 つは動的メタオブジェクトを使用したシナリオです。いずれのシナリオでも ApplicationX を使用します。ApplicationX を使用すると、会社連絡先情報の作成、更新、削除時にその情報を交換できます。シナリオで作成するビジネス・オブジェクト `Sample_WebSphereMQ_LegacyContact` は、ApplicationX からのメッセージに定義されたフィールドと一致します。ApplicationX が送受信するメッセージのフォーマットは、(WebSphere Business Integration Server Express 開発キットに付属している) 区切りデータ・ハンドラーに準拠しています。

また、このチュートリアルではポート・コネクタ・リポジトリを使用します。ポート・コネクタ・リポジトリは WebSphere アダプターをインストールすればそのコンポーネントとしてインストールされます。ポート・コネクタはコネクタの定義のみから構成され、基本となるコードは存在しないため、シミュレーション・シナリオに適しています。

始動した Adapter for WebSphere MQ は、ApplicationX が入力キューに送付した連絡メッセージを検索します。アダプターは区切りデータ・ハンドラーを使用することにより、これらのメッセージを `Sample_WebSphereMQ_LegacyContact` ビジネス・オブジェクトに変換し、統合ブローカーに送達します。Test Connector (やはり、WBI をインストールすると組み込まれているコンポーネント) を使用することにより、ポート・コネクタをシミュレートし、Adapter for WebSphere MQ が送付したビジネス・オブジェクトを検索し、属性を確認することができます。データを変更してから、メッセージを統合ブローカーに再送達します。ここからメッセージは

Adapter for WebSphere MQ に送信され、メッセージに変換され、アダプターの出力キュー (ApplicationX の入力キュー) に送達されます。このチュートリアルでは、アダプターは WebSphere MQ Integration Broker 用に構成されていますが、チュートリアルを実行するためにこのブローカーを実際にインストールおよび構成する必要はありません。

始める前に

チュートリアルを開始する前に、以下のことを確認してください。

- IBM WebSphere 製品がインストールされ、その運用経験をもっていること。
- WebSphere MQ 5.1 またはそれ以上がインストールされていること。
- WebSphere MQ client libraries for Java がインストールされていること。
- Adapter for WebSphere MQ がインストールされていること (構成に関する説明はこのチュートリアルに記載されています)。
- WebSphere MQ アダプター・キュー・マネージャーの名前が `crossworlds.queue.manager` (インストール時のデフォルト値) であること。キュー・マネージャーが他の名前の場合には、この文書で `crossworlds.queue.manager` と記載された個所を使用しているキュー・マネージャーの名前で置き換えてください。

環境のセットアップ

このセクションでは、チュートリアルを使用して作業できる環境の準備の仕方について説明します。後出の `sample_folder` は、サンプルがあるフォルダーを指します。ビジネス・オブジェクト・リポジトリは `sample_folder` 内に `.xsd` ファイルとして提供されています。

1. **キューの定義** このチュートリアルでは、キュー・マネージャーに 6 つのキューが定義されていることが必要です。必要なキューを作成するには、コマンド行から `RUNMQSC crossworlds.queue.manager` と入力して、以下のコマンドを発行します。

- `DEFINE QL('MQCONN.IN')`
- `DEFINE QL('MQCONN.IN_PROGRESS')`
- `DEFINE QL('MQCONN.ERROR')`
- `DEFINE QL('MQCONN.ARCHIVE')`
- `DEFINE QL('MQCONN.REPLY')`
- `DEFINE QL('LEGACYAPP.IN')`

次に、WMQI ブローカーを構成するために WebSphere MQ アダプターおよびポート・コネクターが必要とするキューを以下のように定義します。

- `DEFINE QL('WebSphereMQConnector/ADMININQUEUE')`
- `DEFINE QL('WebSphereMQConnector/ADMINOUTQUEUE')`
- `DEFINE QL('WebSphereMQConnector/DELIVERYQUEUE')`
- `DEFINE QL('WebSphereMQConnector/FAULTQUEUE')`
- `DEFINE QL('WebSphereMQConnector/REQUESTQUEUE')`
- `DEFINE QL('WebSphereMQConnector/RESPONSEQUEUE')`

- DEFINE QL('WebSphereMQConnector/SYNCHRONOUSREQUESTQUEUE')
 - DEFINE QL('WebSphereMQConnectorSYNCHRONOUSRESPONSEQUEUE')
 - DEFINE QL('PortConnector/ADMININQUEUE')
 - DEFINE QL('PortConnector/ADMINOUTQUEUE')
 - DEFINE QL('PortConnector/DELIVERYQUEUE')
 - DEFINE QL('PortConnector/FAULTQUEUE')
 - DEFINE QL('PortConnector/REQUESTQUEUE')
 - DEFINE QL('PortConnector/RESPONSEQUEUE')
 - DEFINE QL('PortConnector/SYNCHRONOUSREQUESTQUEUE')
 - DEFINE QL('PortConnector/SYNCHRONOUSRESPONSEQUEUE')
2. **アダプターの構成** Connector Configurator Express を使用して、*sample_folder*\WebSphereMQConnector.cfg を開きます。Connector Configurator Express の使用方法に関する詳細情報は、83 ページの『付録 B. Connector Configurator』を参照してください。コネクタ固有プロパティの詳細については、24 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。アダプターをまだ構成していない場合は、ご使用のシステムに応じた「インストール・ガイド」の説明に従って構成してください。さらに、以下に示す値と一致するようにアダプターの構成プロパティを確認または変更します。

- Broker Type このプロパティを WMQI に設定します。
- Repository Directory このプロパティを *sample_folder* ディレクトリに設定します。

以下のコネクタ固有プロパティを設定します。

- ConfigurationMetaObject このプロパティを Sample_WebSphereMQ_MO_Config に設定します。
 - DataHandlerConfigMO このプロパティを Sample_WebSphereMQ_MO_DataHandler に設定します。
 - DataHandlerMimeType このプロパティを text/delimited に設定します。
 - DataHandlerClassName このプロパティを com.crossworlds.DataHandlers.text.delimited に設定します。
 - ErrorQueue このプロパティを queue://crossworlds.queue.manager/MQCONN.ERROR に設定します。
 - InProgressQueue このプロパティを queue://crossworlds.queue.manager/MQCONN.IN_PROGRESS に設定します。
 - InputQueue このプロパティを queue://crossworlds.queue.manager/MQCONN.IN に設定します。
 - hostname このプロパティをご使用のマシン名に設定します。
 - port このプロパティを 1414 に設定します。
 - channel このプロパティを CHANNEL1 に設定します。
 - UnsubscribedQueue このプロパティを queue://crossworlds.queue.manager/MQCONN.UNSUBSCRIBED に設定します。
3. **ポート・コネクタの構成** Connector Configurator Express を使用して、以下の標準プロパティを設定します。

- **Broker Type** このプロパティを `WMQI` に設定します。
 - **Repository Directory** このプロパティを `sample_folder` ディレクトリーに設定します。
 - **RequestQueue** このプロパティを `WebSphereMQConnector/DELIVERYQUEUE` に設定します (`WebSphere MQ` アダプターの `DeliveryQueue` プロパティ値)。
 - **DeliveryQueue** このプロパティを `WebSphereMQConnector/REQUESTQUEUE` に設定します (`WebSphere MQ` アダプターの `RequestQueue` プロパティ値)。
4. **ビジネス・オブジェクトのサポート** ビジネス・オブジェクトを使用するには、まずアダプターがビジネス・オブジェクトをサポートする必要があります。
Connector Configurator Express を使用して、`WebSphere MQ` アダプターの「サポートされているビジネス・オブジェクト」タブをクリックして、表 10 に記載されているビジネス・オブジェクトを追加します。「メッセージ・セット ID」をサポートされているそれぞれのビジネス・オブジェクトごとに固有な値に設定します。

表 10. JMS アダプターについてサポートされるサンプル・ビジネス・オブジェクト

ビジネス・オブジェクト名	メッセージ・セット ID
Sample_WebSphereMQ_MO_Config	1
Sample_WebSphereMQ_MO_DataHandler	2
Sample_WebSphereMQ_LegacyContact	3

Connector Configurator Express を使用して、`sample_folder` 内にあるポート・コネクタ定義 `PortConnector.cfg` を開きます。次に、表 11 に記載されたサポートされるビジネス・オブジェクトとメッセージ・セット ID を追加します。

表 11. ポート・コネクタについてサポートされるサンプル・ビジネス・オブジェクト

ビジネス・オブジェクト名	メッセージ・セット ID
Sample_WebSphereMQ_LegacyContact	1

5. **メタオブジェクトの構成** `WebSphere MQ` キュー・マネージャーの名前が `crossworlds.queue.manager` でない場合は、`Sample_WebSphereMQ_MO_Config` ビジネス・オブジェクトのデフォルト属性の「AppSpecificInfo」フィールドの URI を更新する必要があります。

6. **コネクタ開始スクリプトの作成または更新**

Windows の場合

- Adapter for `WebSphere MQ` のショートカットのプロパティを開きます。
- ターゲットの最後の引数として、`-c` の後ろに `<WebSphereMQConnector.cfg` ファイルの絶対パスおよびファイル名> を続けたものを追加します。例:

```
-cProduct_Dir%connectors%WebSphereMQ%samples%  
LegacyContact%WebSphereMQConnector.cfg
```

Linux:

- ファイル `Product_Dir/bin/connector_manager_WebSphereMQ` を開きます。
- `AGENTCONFIG_FILE` プロパティを `-c` の後ろに `<WebSphereMQConnector.cfg` ファイルの絶対パスおよびファイル名> を続けた値に設定します。例:

```
AGENTCONFIG_FILE=-cProduct_Dir/connectors/WebSphereMQ/samples/  
LegacyContact/WebSphereMQConnector.cfg
```

シナリオの実行

シナリオを実行する前に、以下の手順を実行します。

1. **Adapter for WebSphere MQ** がまだ稼働していない場合は始動します。
2. **Visual Test Connector** がまだ稼働していない場合は始動します。

静的メタオブジェクトを使用したシナリオ

チュートリアルはこのセクションでは、静的メタオブジェクトを使用したシナリオについて説明します。静的メタオブジェクトの詳細については、39 ページの『静的メタオブジェクトの作成の概要』を参照してください。

1. **ポート・コネクターのシミュレート** Visual Test Connector を使用して、ポート・コネクターのプロファイルを定義します。
 - a. 「Visual Test Connector」メニューから「ファイル」->「プロファイルを作成/選択」を選択し、次に、「コネクタ・プロファイル」メニューから「ファイル」->「新規プロファイル」を選択します。
 - b. *sample_folder* 内にあるポート・コネクタ構成ファイル PortConnector.cfg を選択して、Connector Name および Broker Type を構成してから「OK」をクリックします。
 - c. 作成したプロファイルを選択し、「OK」をクリックします。
 - d. 「Visual Test Connector」メニューから、「ファイル」->「接続」を選択してシミュレートを開始します。
2. **要求処理のテスト**
 - a. Test Connector を使用して、ビジネス・オブジェクト Sample_WebSphereMQ_LegacyContact の新規インスタンスを作成します。これを実行するには、**BoType** ドロップダウン・ボックスでビジネス・オブジェクトを選択してから、BOInstance の「作成」を選択します。
 - b. 必要に応じてデフォルト値を変更し、動詞を **Create** に設定して、「ビジネス・オブジェクトを送信」をクリックしてメッセージを送信します。
3. **メッセージ送達の検査** WebSphere MQ Explorer または同様のアプリケーションを使用してキュー `queue://crossworlds.queue.manager/LEGACYAPP.IN` を開き、フォーマットが LC_CR の新規連絡メッセージがアダプターから届いているか確認します。
4. **イベント処理のテスト** メッセージを WebSphere MQ アダプターの入力キューに送信します。注: このステップでは、キューにメッセージを送ることができるユーティリティが必要で、このようなユーティリティが使用できない場合は、WebSphere アダプターの InputQueue プロパティを `queue://crossworlds.queue.manager/LEGACYAPP.IN` に設定します。これにより、アダプターは自身のメッセージをポーリングできます (これが最も容易な方法です)。入力キューにメッセージが入ると、アダプターはこのメッセージに対するポーリングを実行し、これを Sample_WebSphereMQ_LegacyContact ビジネス・オブジェクトに変換しようとします。アダプターにメッセージのポーリングを実行させるために重要なことは、メッセージ・フォーマットが、メタオブジェクト Sample_WebSphereMQ_MO_Config 内の Sample_WebSphereMQ_LegacyContact ビジネス・オブジェクトに関連付けられた値と等しいことです。このシナリオの場合、フォーマットは LC_CR です。アダプターは、着信メッセージ・フォーマ

ットを LC_CR であると認識すると、データ・ハンドラーを使用して、動詞 Create 付きビジネス・オブジェクト Sample_WebSphereMQ_LegacyContact にメッセージを変換します。その後、この新しく作成されたビジネス・オブジェクトは Test Connector に送達されます。

5. **メッセージ送達の確認** 上記のステップがすべて正常に実行された場合には、適切なサンプル・シナリオが得られ、このシナリオにより WebSphere MQ アダプターがメッセージを検索し、これらのメッセージを Sample_WebSphereMQ_LegacyContact ビジネス・オブジェクトに変換し、さらに、逆に Sample_WebSphereMQ_LegacyContact ビジネス・オブジェクトを連絡メッセージに変換することが可能になります。

動的メタオブジェクトを使用したシナリオ

このシナリオでは、動的メタオブジェクトを使用して、静的メタオブジェクトのシナリオに定義された各種のキューにビジネス・オブジェクトを転送する方法について説明します。動的メタオブジェクトの詳細については、41 ページの『動的子メタオブジェクトの作成の概要』を参照してください。このシナリオの前提条件については、104 ページの『始める前に』を参照してください。さらに、107 ページの『静的メタオブジェクトを使用したシナリオ』の説明に従ってポート・コネクターをインストールおよび構成する必要があります。以下のステップでは、Sample_WebSphereMQ_LegacyContact の子メタオブジェクトの属性を作成します。特に、この子メタオブジェクトの出力キュー値を変更することにより、Sample_WebSphereMQ_LegacyContact ビジネス・オブジェクトを新規のキューに転送します。

1. **動的メタオブジェクト属性の識別** まず、動的メタオブジェクトが設定された属性を識別するために、アプリケーション固有情報を追加する必要があります。Sample_WebSphereMQ_LegacyContact で、cw_mo_conn=DynMO をアプリケーション固有情報に追加します。これにより属性が識別されます。
2. **属性の追加** Business Object Designer Express を使用して、以下の手順を実行します。
 - a. *sample_folder* から Sample_WebSphereMQ_DynMO_Config.xsd および Sample_WebSphereMQ_LegacyContact.xsd を開きます。
 - b. 「Sample_WebSphereMQ_LegacyContact」ウィンドウで、名前が DynMO でタイプが Sample_WebSphereMQ_DynMO_Config の属性を追加します。
3. **新しいターゲット・キューの定義** WebSphere MQ の一時キュー REROUTE.IN を定義します。これは、動的メタオブジェクトによる Sample_WebSphereMQ_LegacyContact ビジネス・オブジェクトの転送先です。必要なキューを作成するには、コマンド行から RUNMQSC crossworlds.queue.manager と入力して、以下のコマンドを発行します。

```
DEFINE QL('REROUTE.IN')
```
4. **Adapter for WebSphere MQ がまだ稼働していない場合は始動します。**
5. **Visual Test Connector がまだ稼働していない場合は始動します。**
6. **ポート・コネクターのシミュレート** (静的メタオブジェクトを使用したシナリオでこのステップをすでに実行している場合は、この作業をスキップして次の作業に進んでください。) Visual Test Connector を使用して、ポート・コネクターのプロファイルを定義します。

- a. 「Visual Test Connector」メニューから「ファイル」->「プロファイルを作成/選択」を選択し、次に、「コネクタ・プロファイル」メニューから「ファイル」->「新規プロファイル」を選択します。
 - b. Samples ディレクトリー内にあるポート・コネクタ構成ファイル PortConnector.cfg を選択して、Connector Name および Broker Type を構成してから「OK」をクリックします。
 - c. 作成したプロファイルを選択し、「OK」をクリックします。
 - d. 「Visual Test Connector」メニューから、「ファイル」->「接続」を選択してシミュレートを開始します。
7. 親ビジネス・オブジェクトおよび子メタオブジェクトのインスタンスを作成
Visual Test Connector を使用して、以下の手順を実行します。
- a. ビジネス・オブジェクト Sample_WebSphereMQ_LegacyContact の新規インスタンスを作成し、必要に応じてデフォルト値を変更します。
 - b. DynMO 属性を右マウス・ボタン・クリックして、そのインスタンス Sample_WebSphereMQ_DynMO_Config を作成します。
8. 新しいターゲット・キューの設定
- a. DynMO 属性の横にある + 符号をクリックして、この属性を展開します。
 - b. outputQueue という名前の属性に、ターゲット・キューの名前を入力します。このシナリオの場合、ターゲット・キューは REROUTE.IN です。
queue://<queue manager>/REROUTE.IN?targetClient=1 など完全な URI を入力します。
9. ビジネス・オブジェクトの送信 「ビジネス・オブジェクトを送信」をクリックします。
10. メッセージ送達の確認 WebSphere MQ Explorer または同様のアプリケーションを使用してキュー queue://<queue manager>/REROUTE.IN を開き、新規連絡メッセージがアダプターから届いているか確認します。新しいメッセージが WebSphere MQ アダプターから REROUTE.IN というキューに届いていれば、転送が成功したことを示しています。WebSphere MQ で各種のキューを作成し、ビジネス・オブジェクトにこれらのキュー名を指定して送信します。その場合、キュー名はビジネス・オブジェクトのそれぞれの動的メタオブジェクトに指定します。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。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 Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性があります。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認できません。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。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。本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。著作権使用許諾: 本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほめめかしたり、保証することはできません。この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

プログラミング・インターフェース情報は、プログラムを使用してアプリケーション・ソフトウェアを作成する際に役立ちます。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
CICS
CrossWorlds
DB2
DB2 Universal Database
IMS
i5/OS
Informix
iSeries
Lotus
Lotus Domino
Lotus Notes
MQIntegrator
MQSeries
MVS
OS/400
Passport Advantage
SupportPac
WebSphere
z/OS

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

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

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

Linux は、Linus Torvalds の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

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



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

索引

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

[ア行]

アーカイブ 10
アーカイブ・キュー 10
アダプターのインストール 17
アダプター・プラットフォーム 2
アプリケーション・タイムアウト 52
イベント処理
 同期 7
 問題 55
イベント通知 6
イベント・デリバリー 5
 保証付き 10
インストール
 アダプターと関連ファイル 21
 統合ブローカー 21
 Linux での検証 22
 OS/400 および i5/OS での検証 23
 Windows での検証 22
インストール済みファイルの構造 21
エラー処理 52
オペレーティング・システムの要件 2

[カ行]

関連文書 v
キュー
 構成 19
キューの Uniform Resource Identifier 32
コネクタ
 構成プロパティ 23
 始動 46
 始動スクリプト 46
 停止 47
 プロパティ
 構成 17
コネクタ固有の構成プロパティ 24
コネクタの始動 46
コネクタの停止 47
コラボレーション 3

[サ行]

作成
 静的メタオブジェクト 40

作成 (続き)
 動的子メタオブジェクト 41
 複数コネクタ・インスタンス 30
始動スクリプト
 コネクタの場合 46
 変更 19
シナリオ 107
 静的メタオブジェクト 107
 動的メタオブジェクト 108
静的メタオブジェクト 34
 サンプル 39
双方向言語サポート 15
双方向スクリプト・データ 3
ソフトウェア前提条件 2

[タ行]

タイムアウト 52
チュートリアル 103
 環境のセットアップ 104
 シナリオの実行 107
データ・ハンドラー
 構成 19
 入力キューとのマップ 40
データ・ハンドラーの変換 53
動詞の処理 11
動的子メタオブジェクト
 および JMS ヘッダー 43
 作成 41
動的メタオブジェクト 34
トラブルシューティング 55
トレース 53

[ハ行]

ビジネス・オブジェクト 49
 アンサブスクリाइブ 52
ビジネス・オブジェクト要求 11
ビジネス・オブジェクト・プロパティ
 サンプル 50
標準構成プロパティ 57
フィードバック・コード
 カスタム 16
複数コネクタ・インスタンス 30
プラットフォーム 2
プロパティ
 コネクタ固有 24
 標準 57
保証付きイベント・デリバリー 10

[マ行]

メタオブジェクト
 静的
 作成 40
 サンプル 39
 動的 18
 動的子 41
メタオブジェクトの構成 33
メタオブジェクト・プロパティ 34
メッセージ
 記述子 18
 検索 6
 要求 4
 リカバリー 9
問題
 始動時 55

[ヤ行]

要求
 送信および取得通知 17
 通知なしでの送信 17
 ビジネス・オブジェクト 11

[ラ行]

リカバリー、メッセージの 9
ロケール依存データ 2

[数字]

2 バイト文字セット 2

A

ApplicationPassword プロパティ 25
ApplicationUserName 25
ArchiveQueue 26

B

BiDi.Application 標準プロパティ 15

C

CCSID プロパティ 26
Channel プロパティ 26
ConfigurationMetaObject 26

Connector Configurator 23, 83
Create、Update、および Delete 11

D

DataHandlerClassName 26
DataHandlerConfigMO 26
DataHandlerMimeType 26
DefaultVerb プロパティ 26
DeliveryTransport 標準プロパティ 10

E

EnableMessageProducerCache 27
ErrorQueue プロパティ 27

F

FaultQueue 11
FeedbackCodeMappingMO 27

H

HostName プロパティ 28

I

InDoubtEvents プロパティ 28
InProgressQueue プロパティ 29
InputQueue プロパティ 28

J

Java Message Service 3
Java ランタイム環境
Unicode 文字コード・セット 3
JMS 3
トランスポート 10
プロパティ 43

M

MQMDs 18

P

PollQuantity プロパティ 29
Port プロパティ 29

R

ReplyToQueue プロパティ 29
ReplyToQueuePollFrequency 29

Retrieve 動詞、Exists 動詞、および
Retrieve By Content 動詞 16

S

SessionPoolSizeForRequests 29
SSL 2
SSL (Secure Sockets Layer) 2

U

UnsubscribedQueue プロパティ 29
URI 32
UseDefaults プロパティ 30

W

WebSphere Business Integration コラボレ
ーション 3
WebSphere MQ メッセージ 3



Printed in Japan