IBM WebSphere Business Integration Adapters

Adapter for WebSphere MQ ユーザーズ・ガイド

Adapter バージョン 2.8.x

IBM WebSphere Business Integration Adapters

Adapter for WebSphere MQ ユーザーズ・ガイド

Adapter バージョン 2.8.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、 135 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 2.8.x に適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： IBM WebSphere Business Integration Adapters
Adapter for WebSphere MQ User Guide
Adapter Version 2.8.x

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2005.12

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

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

目次

まえがき

本書について

IBM® WebSphere® Business Integration Adapter ポートフォリオは、優れた e-business テクノロジー、エンタープライズ・アプリケーション、レガシー・システム、およびメインフレーム・システムへの統合コネクティビティーを提供します。この製品セットには、ビジネス統合のコンポーネントをカスタマイズ、作成、および管理するためのツールやテンプレートが含まれています。

本書に含まれる内容

この資料では、この IBM WebSphere Business Integration Adapter のインストール、コネクター・プロパティー構成、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。

本書に含まれない内容

この文書は、サーバーのロード・バランシング、アダプター処理スレッドの数、最大および最小スループット、許容度しきい値などの、配置のメトリックおよびキャパシティー・プランニングについては説明しません。

このような問題は、すべてのカスタマーの配置に固有であり、アダプターが配置される環境内またはそれに近い環境で測定する必要があります。使用する配置サイトの構成についての質問や、特定の構成を前提とした、この種のメトリックの計画と評価の詳細については、IBM サービス技術員にお問い合わせください。

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

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

関連資料は以下のサイトからインストールできます。

- アダプターの一般情報が必要な場合、アダプターを WebSphere Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) と共に使用する場合、およびアダプターを WebSphere Application Server と共に使用する場合は、以下のサイトにある IBM WebSphere Business Integration Adapters インフォメーション・センターを参照してください。 http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- アダプターを WebSphere InterChange Server と共に使用する場合は、以下のサイトにある IBM WebSphere InterChange Server インフォメーション・センターを参照してください。 http://www.ibm.com/websphere/integration/wicserver/infocenter
  http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- WebSphere Message Brokers の詳細については、以下のサイトを参照してください。
  http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- WebSphere Application Server の詳細については、以下のサイトを参照してください。
  http://www.ibm.com/software/webservers/appserv/library.html

これらのサイトには、資料をダウンロード、インストール、および表示するための簡単な指示が掲載されています。

表記上の規則

本書では、以下の規則を使用します。

`Courier フォント`	コマンド名、ファイル名、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のリテラル値を示します。
太字	初出語を示します。
イタリック、*イタリック*	変数名または相互参照を示します。
青のアウトライン	青のアウトラインは、マニュアルをオンラインで表示するときにのみ見られるもので、相互参照用のハイパーリンクを示します。 アウトラインの内側をクリックすることにより、参照先オブジェクトにジャンプできます。
{ }	構文の記述行の場合、中括弧 {} で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
[ ]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプション・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・パスに円記号 (¥) を使用します。UNIX(R) システムの場合には、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 *$text*	% 記号で囲まれたテキストは、Windows (R) の text システム変数またはユーザー変数の値を示します。UNIX 環境の場合、これに相当する表記は *$text* になります。これは、*text* UNIX 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。

本リリースの新機能

リリース 2.8.x での新機能

今回のリリースでは、AIX 5.3 プラットフォームのサポートを追加しています。この点と、その他のハードウェア要件およびソフトウェア要件については、http://www.ibm.com/support/docview.wss?uid=swg27006249 を参照してください。この URL で提供する情報は、今回のリリースの時点で本書から削除されている『ブローカーとの互換性』および『アダプター・プラットフォーム』セクションの情報に替わるものです。

アダプターは、IBM Tivoli License Manager (ITLM) をサポートするようになりました。

アダプターは、イベント・ポーリングに複数のスレッドを使用できるようになりました。Connector Configurator 内の WorkerThreadCount プロパティーでスレッドの最大数を指定します。詳細については、31 ページの『WorkerThreadCount』を参照してください。

新規のコネクター固有のプロパティー DataHandlerPoolSize によって、それぞれのデータ・ハンドラー・タイプごとにインスタンス・プールを作成できます。この機能拡張によって、アダプターは、後で使用するために構成済みデータ・ハンドラーのインスタンスをプールすることができます。詳細については、27 ページの『DataHandlerPoolSize』を参照してください。

新規のコネクター固有のプロパティー SecurityExitClassName および SecurityExitInitParam を使用して、カスタム・セキュリティー出口を呼び出すようにアダプターを構成できるようになりました。詳細については、SecurityExitClassName および SecurityExitInitParam を参照してください。

アダプターのデータ・エンコード (テキストをサポート) は、バイナリーおよび JMS オブジェクトのメッセージ・タイプをサポートするようになりました。メタオブジェクトの DataEncoding プロパティーで、テキスト (デフォルトのデータ・エンコード)、バイナリー、またはオブジェクトを指定します。詳細については、40 ページの『バイナリーおよびオブジェクト・メッセージ用の DataEncoding』を参照してください。

リリース 2.7.x での新機能

2004 年 9 月更新。Adapter バージョン 2.7.x に対応して、本書には以下の新情報または訂正情報が含まれています。

今回のリリースでは、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。

今回のリリースでは、以下のプラットフォームまたはアップデートのサポートが追加されています。

- Microsoft Windows 2000 (Professional、Server、または Advanced Server) (Service Pack 4 を適用)
- Microsoft Windows 2003 (Standard Edition または Enterprise Edition)
- Solaris 8 (2.8) (2004 年 2 月 11 日以降の日付の Solaris Patch Cluster を適用)
- Solaris 9 (2.9) (2004 年 2 月 11 日以降の日付の Solaris Patch Cluster を適用)。このアダプターは、64 ビットのプラットフォームで 32 ビット JVM をサポートします。
- AIX 5.1 (Maintenance Level 4 を適用)
- AIX 5.2 (Maintenance Level 1 を適用)
  このアダプターは、64 ビットのプラットフォームで 32 ビット JVM をサポートします。
- HP-UX 11i (11.11) (June 2003 GOLDBASE11i バンドルおよび June 2003 GOLDAPPS11i バンドルを適用)
- Red Hat Enterprise Linux AS 3.0 (アップデート 1)、ES 3.0（アップデート 1)、または WS 3.0 (アップデート 1)
- SUSE Linux Enterprise Server x86 8.1 (SP3 を適用)
- SUSE Linux Standard Server x86 8.1 (SP3 を適用)
- すべてのオペレーティング・システム環境で、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 (IBM JDK 1.4.2 for Windows 2000) が必要です。

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

今回のリリースでは、アダプターがキャッチした例外で printStackTrace() をダンプするためのトレース・レベル 5 の使用をサポートします。

OutputFormat メタオブジェクト・プロパティーの名前の長さは 16 文字までという制限が、文書化されました。

リリース 2.6.x での新機能

2 つのコネクター固有のプロパティー EnableMessageProducerCache および SessionPoolSizeForRequests が追加されました。詳細については、25 ページの『コネクター固有のプロパティー』を参照してください。

同期要求に対する応答メッセージを処理する際、コネクターはフィードバック・コード MQFB_NONE (設定されていない場合のデフォルトのフィードバック・コード) を VALCHANGE として解釈します。詳細については、12 ページの『同期デリバリー』を参照してください。

バージョン 2.6.x から、アダプターは Solaris 7 でサポートされなくなりました。そのため、このプラットフォーム・バージョンに関する記述は本書から削除されました。

リリース 2.5.x での新機能

コネクターは、以下のプラットフォーム上で実行されます。

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

バージョン 2.5.0 からは、Adapter for WebSphere MQ は Microsoft Windows NT ではサポートされなくなりました。

アダプターのインストール情報は本書から移動しました。この情報の新規掲載場所については 2 章を参照してください。

リリース 2.4.x での新機能

アダプターは、WebSphere Application Server を統合ブローカーとして使用できるようになりました。

コネクターは、以下のプラットフォーム上で実行されます。

- Microsoft Windows NT(R) 4.0 Service Pack 6A または Windows 2000
- Solaris 7、8 または AIX 5.1、5.2 または HP UX 11.i

リリース 2.3.x での新機能

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

データ・ハンドラーを入力キューと関連付けることができるようになりました。詳細については、 42 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。

保証付きイベント・デリバリー機能が拡張されました。詳細については、 10 ページの『保証付きイベント・デリバリー』を参照してください。

リリース 2.2.x での新機能

InProgress キューは不要になりました。使用不可に設定できます。詳細については、 29 ページの『InProgressQueue』を参照してください。

コネクターは、WebSphere MQ 5.1、5.2、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ーをサポートします。詳細については、 2 ページの『アダプターの依存関係』を参照してください。

コネクターに、ビジネス・オブジェクト処理のための UseDefaults プロパティーが追加されました。詳細については、 31 ページの『UseDefaults』を参照してください。

データ・ハンドラーが明示的にビジネス・オブジェクトに対して動詞を割り当てていない場合、コネクターがデフォルトの動詞を適用できるようになりました。詳細については、 27 ページの『DefaultVerb』を参照してください。

ReplyToQueue は、ReplyToQueue コネクター・プロパティーではなく動的子メタオブジェクトを介して指定できるようになりました。詳細については、 45 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。

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えます。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 JMS 機能は同期要求処理にのみ適用されます。詳細については、 12 ページの『同期デリバリー』を参照してください。

リリース 2.1.x での新機能

コネクターは国際化されました。詳細については、 2 ページの『ロケール依存データ』と 59 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

本書では、このアダプターを InterChange Server と共に使用するための情報を提供します。

注: 保証付きイベント・デリバリー機能を使用するには、WebSphere InterChange Server のリリース 4.1.1.2 をインストールする必要があります。

リリース 1.5.x での新機能

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

- WebSphere MQ 専用のアプリケーション・コンポーネント
- サンプル・ビジネス・オブジェクト
- IBM WebSphere Adapter フレームワーク。以下から構成されています。
  - コネクター・フレームワーク
  - 開発ツール (Business Object Designer と IBM CrossWorlds System Manager を含む)
  - API (CDK を含む)

このマニュアルでは、このアダプターを InterChange Server と共に使用するための情報を提供します。

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

AIX 4.3.3 パッチ・レベル 9 で、コネクターが使用可能になりました。

リリース 1.4.x での新機能

本書では、コネクターのバージョン 1.4.x における以下の新機能または機能変更について説明します。

- 以前のバージョンの WebSphere MQ 対応コネクターでは、WebSphere MQ メッセージと CrossWorlds ビジネス・オブジェクト間のデータ変換に使用するデータ・ハンドラーが、コネクター・プロパティー DataHandlerConfigMO および DataHandlerMimeType によって決定されました。このため、異なるデータ・フォーマットを処理するためには、コネクターのインスタンスが複数必要でした。リリース 1.4.x では、コネクターの静的メタオブジェクトまたはビジネス・オブジェクトの動的子メタオブジェクトの要求内で、これらのプロパティーをオプションで指定できるようになりました。詳細については、34 ページの『メタオブジェクトの構成』を参照してください。
- DataEncoding メタオブジェクト属性 (静的または動的) が追加されたことにより、メッセージ・タイプ (テキストまたはバイナリー) とエンコード・タイプを指定できるようになりました。詳細については、34 ページの『メタオブジェクトの構成』を参照してください。

リリース 1.3.x での新機能

本書では、コネクターのバージョン 1.3.x における以下の新機能または機能変更について説明します。

- 外部アプリケーションからコネクターに対して発行された同期要求のサポート。詳細については、7 ページの『同期イベント処理』を参照してください。
- CollaborationName プロパティーがメタオブジェクトに追加されました。34 ページの『メタオブジェクトの構成』を参照してください。
- DoNotReportBusObj プロパティーがメタオブジェクトに追加されました。34 ページの『メタオブジェクトの構成』を参照してください。

第 1 章 概要

WebSphere MQ 用のコネクターは、WebSphere Business Integration Adapter for WebSphere MQ のランタイム・コンポーネントの 1 つです。このコネクターを使用すると、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、コネクター・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・アーキテクチャーについて説明します。

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

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

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

統合ブローカーとコネクターの関係についての詳細は、「*IBM WebSphere InterChange Server システム管理ガイド*」を参照してください。

注: すべての WebSphere Business Integration アダプターは、統合ブローカーと連携して動作します。WebSphere MQ 用コネクターは、以下のソフトウェアと連携して動作します。

- InterChange Server 統合ブローカー。詳細は「*テクニカル入門 (IBM WebSphere InterChange Server)*」を参照してください。
- WebSphere Application Server (WAS) 統合ブローカー。詳細は、「*アダプター実装ガイド (WebSphere Application Server)*」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。

- 『前提条件』
- 『アダプターの依存関係』
- 『ロケール依存データ』
- 3 ページの『Common Event Infrastructure』
- 3 ページの『Application Response Measurement』

前提条件

ブローカーの互換性、サポートされるプラットフォームについて詳しくは、http://www.ibm.com/support/docview.wss?uid=swg27006249 を参照してください。

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。

- コネクターは、WebSphere MQ 5.1、5.2[1]、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ーをサポートします。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 Secure Sockets Layer (SSL) 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とって適切な WebSphere MQ ソフトウェア・バージョンについては、使用プラットフォーム (Windows または UNIX) のインストール・ガイドを参照してください。

ロケール依存データ

コネクターは、2 バイト文字セットをサポートし、指定された言語でメッセージ・テキストを配送できるように、国際化されています。コネクターが、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。

このアダプターは、Windows 環境で実行される場合は、アラビア語およびヘブライ語の双方向スクリプト・データの処理をサポートします。双方向処理は、Windows 以外の環境ではサポートされません。双方向機能を使用するためには、標準の双方向プロパティー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ー』を参照してください。

Java 仮想マシン (JVM) 内の Java ランタイム環境では、データは Unicode 文字コード・セットで表現されます。 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

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。

み込まれています。WebSphere Business Integration システムのコンポーネントの大部分は Java で作成されています。したがって、ほとんどのインテグレーション・コンポーネント間で、文字変換を行わずにデータを転送できます。

エラー・メッセージや通知メッセージを国や地域に応じた適切な言語で記録するには、該当する環境の Locale 標準構成プロパティーを設定します。構成プロパティーの詳細については、59 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

Common Event Infrastructure

このアダプターは、IBM が提供する Common Event Infrastructure との互換性があります。Common Event Infrastructure は、イベントを生成するその他の IBM WebSphere アプリケーションとのインターオペラビリティーを許可するイベント管理の標準です。Common Event Infrastructure サポートが使用可能である場合、アダプターが生成したイベントは、Common Event Infrastructure と互換性のある別のアプリケーションが受信 (または使用) できます。

詳しくは、本書の Application Response Management に関する付録を参照してください。

Application Response Measurement

このアダプターは、Application Response Measurement (ARM) アプリケーション・プログラミング・インターフェース (API) との互換性があります。この API によって、アプリケーションの可用性、サービス・レベル・アグリーメント、およびキャパシティー・プランニングを管理することができます。ARM を備えたアプリケーションは、IBM Tivoli(R) Monitoring for Transaction Performance に参加することができ、これにより、トランザクションのメトリックに関するデータの収集および検討が可能です。

詳しくは、本書の Application Response Measurement に関する付録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、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。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 API で、保証付きイベント・デリバリーも可能になります。

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 メッセージを送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・エントリー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、コネクター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ーを定義していない場合、デフォルトでは、コネクターは静的メタオブジェクトの値を調べます。1 つの静的コネクター・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。

コネクター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。コネクターは、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、コネクターに対し、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、コネクターは、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、コネクターはそのビジネス・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、gotApplEvents() メソッドを使用して InterChange Server にデリバリーします。

アプリケーションとコネクター間の通信方法

コネクターは、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、コネクターはそのビジネス・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、コネクター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。

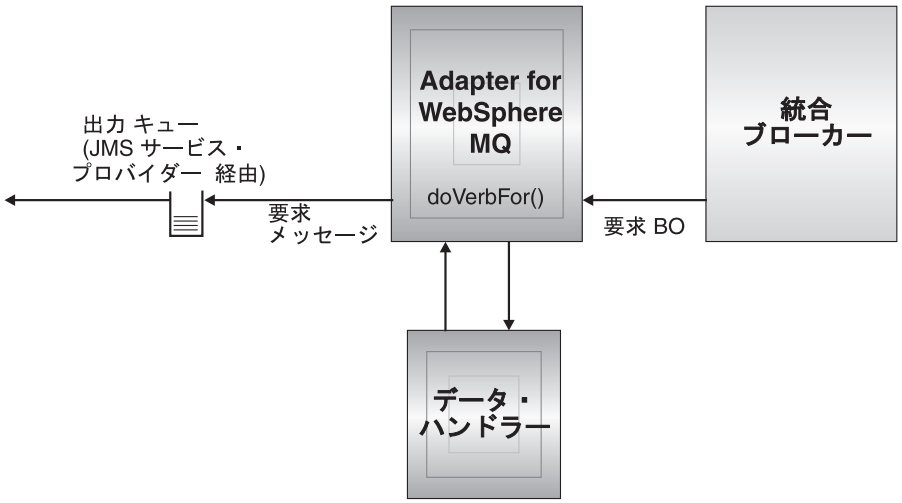

図 1. アプリケーションとコネクターの間の通信方法: メッセージ要求

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。コネクターは最初に、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、コネクター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ーが反映されます。次に、コネクターは、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、`gotApplEvents()` メソッドがビジネス・オブジェクトを InterChange Server 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。

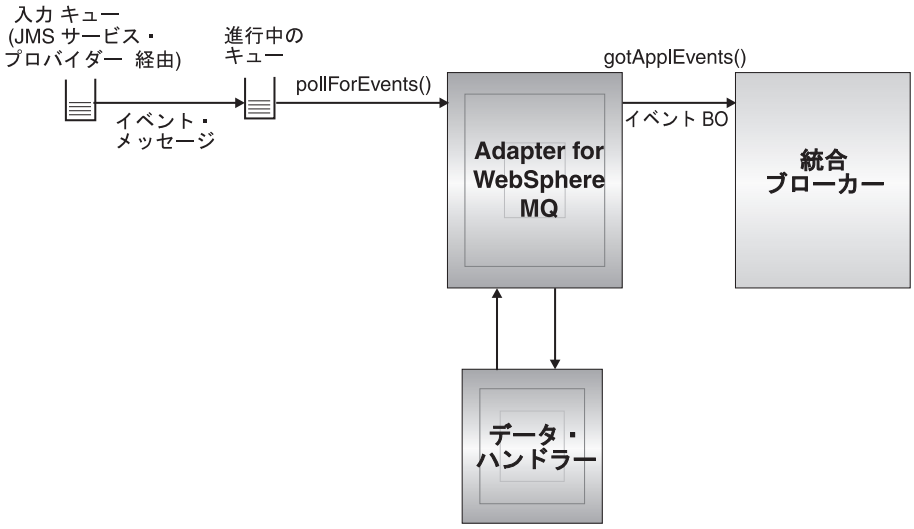

図 2. アプリケーションとコネクターの間の通信方法: イベント・デリバリー

イベント処理

コネクターは、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に発生します。

検索

コネクターは、`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、コネクターはそれを 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ーの静的オブジェクトで定義されている場合、コネクターは、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、コネクター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、54 ページの『エラー処理』を参照してください。

コネクターは、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、コネクターがビジネス・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できなかった場合に、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、コネクター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

コネクター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戻されず、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、コネクター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) `gotApplEvents()` メソッドによる、InterChange Server 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 対応コネクターは、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、コネクターはこのような要求のビジネス・データを InterChange Server 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、コネクターは、InterChange Server 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。コネクター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、コネクターは、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 対応コネクター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 対応コネクターが肯定確認通知レポートまたは否定確認通知レポート (PAN または NAN) を要求するメッセージを受け取った場合、コネクターはそのメッセージの内容を InterChange Server 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。

表 1 に、コネクターに送信された 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。

表 1. WebSphere MQ に必要なメッセージ構造

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`

表 1. WebSphere MQ に必要なメッセージ構造 (続き)

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	次のいずれか一方、または両方を指定できます。 • 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。 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。 • 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。

7 ページの表 1 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。

9 ページの表 2 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。

表 2. レポートの構造

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
feedback	レポートのタイプ	次のいずれかです。 • 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ー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。
メッセージの本体		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、コネクター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 プロパティーを true に設定することによりオーバーライドできます。 要求を処理できなかった場合、コネクターはメッセージの本体にコネクター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。

リカバリー

コネクターは初期化の際に実行中のキューを調べ、コネクター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。コネクターの構成プロパティー 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。

Fail on startup

fail on startup オプションを指定した場合、コネクターが初期化の際、実行中のキュー内にメッセージを検出すると、コネクターはエラーを記録し、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。

Reprocess

reprocessing オプションを指定した場合、コネクターが初期化の際、実行中のキュー内にメッセージを検出すると、コネクター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、コネクターは入力キューからのメッセージの処理を開始します。

Ignore

ignore 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、コネクターはそれを無視しますが、シャットダウンはしません。

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ー 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、 54 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・プロパティーとなります。

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、コネクターの DeliveryTransport 標準プロパティーを JMS に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 トランスポートを使用し、コネクターと統合ブローカーとの間の以降の通信は、すべてこのトランスポートを介して行われます。 JMS トランスポートにより、メッセージは最終的に宛先に確実に配送されます。JMS トランスポートの役割は、トランザクション・キュー・セッションが開始されると、コミットが発行されるまでメッセージがキャッシュされるようにすることです。障害が発生するかまたはロールバックが発行されると、メッセージは破棄されます。

注: 保証付きイベント・デリバリー機能を使用しないと、コネクターがイベントをパブリッシュして (コネクターが pollForEvents() メソッド内で gotApplEvent() メソッドを呼び出して) から、イベント・レコードを削除してイベント・ストアを更新する (または「イベント通知済み」状況に更新する) までの間に、障害が起こる可能性のある短い時間枠が存在します。この間に障害が発生すると、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 回送信されることになります。

JMS イベント・ストアを使用する JMS 対応コネクター用、または JMS イベント・ストアを使用しない JMS 対応コネクター用に保証付きイベント・デリバリー機能を構成することができます。保証付きイベント・デリバリーを行うようにコネクターを構成するには、「*コネクター開発ガイド (Java 用)*」の説明を参照してください。

コネクター・フレームワークがビジネス・オブジェクトを WebSphere InterChange Server 統合ブローカーに配送できない場合、オブジェクトは (UnsubscribedQueue と ErrorQueue ではなく) FaultQueue に配置されて、状況表示と問題の説明を生成します。 FaultQueue メッセージは MQRFH2 フォーマットで書き込まれます。詳細については、57 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 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。コネクターは、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

コネクターは、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、コネクターはビジネス・オブジェクト・ハンドラーと `doForVerb()` メソッドを使用します。コネクターは、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。コネクターは、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。

重要: 生成されたビジネス・オブジェクトで動詞が設定されていない場合、コネクターは、何らかの動詞を含む、このビジネス・オブジェクトに定義された変換プロパティーを検索します。変換プロパティーが 1 セットのみ検出された

場合、コネクターは指定された動詞を割り当てます。複数のプロパティーが検出された場合は、コネクターは動詞を判別できないため、メッセージの処理が失敗します。

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、コネクターは SUCCESS を戻します。それ以外の場合は FAIL を戻します。

注: コネクターには、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

コネクター固有のプロパティーで 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ーに responseTimeout が存在する場合、コネクターは同期モードで要求を送信します。続いて、コネクターは、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、コネクターは表 3 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。

表 3. メッセージ・ヘッダー

フィールド	説明	値
Format	フォーマット名	変換プロパティー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ー・プロパティー ReplyToQueue 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*

表 3. メッセージ・ヘッダー (続き)

フィールド	説明	値
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。

12 ページの表 3 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。

コネクター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、表 4、表 5、および表 6 に示すようなレポート・メッセージを発行します。

表 4. レポート・メッセージ

フィールド	説明	値
Format	フォーマット名	変換プロパティー内で定義された busObj 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。

表 5. レポート・メッセージ: 動詞

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。

表 6. レポート・メッセージ: WebSphere MQ フィードバック・コード

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

表 6. レポート・メッセージ: *WebSphere MQ* フィードバック・コード *(続き)*

WebSphere MQ フィードバック・コード	等値の応答*
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)

* 詳細については、「*コネクター開発ガイド (Java 用)*」を参照してください。

ビジネス・オブジェクトを処理できる場合、アプリケーションは、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、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 が Request 操作のために最初に発行したのと同じビジネス・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 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 ストリング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*'

メッセージ選択子ストリングは、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} を子コンテナー・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・オブジェクトの内容の双方向変換がサポートされています。コネクターが外部 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・フィルター操作機構を使用して ReplyToQueue で応答メッセージを探す場合、双方向言語サポートも提供されています。このような環境では、Windows 標準形式が WebSphere MQ アプリケーションの双方向形式に変換されるように、双方向変換が適用されます。これは、BiDi.Application 標準構成プロパティーに設定された属性を使用して適用されます。

response_selector が

response_selector=JMSCorrelationID LIKE'*selectorstring*'

という構文で指定された場合、および双方向文字が「*selectorstring*」に表示される場合、アダプターは、「*selectorstring*」を外部アプリケーションの双方向形式に変換するように構成することができます。外部アプリケーションは、アダプターが作成した要求に対する応答メッセージの送信を担当します。これは、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ようなアプリケーション固有の情報のデフォルトの Windows 形式と異なる場合に必要になります。このような場合、BiDi.Application 標準構成プロパティーは、外部アプリケーションの双方向形式で構成する必要があります。これによりアダプターは、メッセージ選択子ストリングを、応答メッセージのフィルター操作に使用される前に変換します。

カスタム・フィードバック・コードの作成: コネクター・プロパティー 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。このプロパティーを使用すると、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 に渡されます。詳細については、28 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。

表 7 に、これらの動詞に対応する応答メッセージを示します。

表 7. 応答メッセージ

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。

コネクター・プロパティーの構成

コネクターには、標準構成プロパティーとコネクター固有の構成プロパティーの 2 種類の構成プロパティーがあります。一部のプロパティー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、一部のプロパティーについては、コネクター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 のコネクター・プロパティーを構成する際には、次のことを確認してください。

- コネクター・プロパティー 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。

- コネクター・プロパティー 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。
- コネクター・プロパティー 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ネルと一致している。
- コネクター・プロパティー 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ーの構成

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ーを構成するには、次の作業を実行します。

- 送信する要求を表しており、コネクター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、40 ページの『静的メタオブジェクトの作成の概要』と 43 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ー ResponseTimeout を -1 に設定します。この設定では、コネクター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、12 ページの『Create、Update、および Delete』、 34 ページの『メタオブジェクトの構成』、および 51 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ーの構成

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ーを構成するには、次の作業を実行します。

- コネクター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ーの構成』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、12 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。 34 ページの『メタオブジェクトの構成』と 51 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。

- 40 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ー 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、25 ページの『コネクター固有のプロパティー』を参照してください。

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。これに対して、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、 43 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、12 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ーに、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) の設定』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方法は 2 つあります。

- コネクター固有のプロパティー 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、25 ページの『コネクター固有のプロパティー』を参照してください。
- または、コネクター固有のプロパティー 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・プロパティー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、47 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

この章では、コネクターのインストール方法および構成方法と、メッセージ・フローをコネクターと共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 対応コネクター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 adapter 製品のインストールの詳細については、以下の WebSphere Business Integration Adapters インフォメーション・センターのサイトにある「*WebSphere Business Integration Adapters のインストール* 」を参照してください。

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

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 UNIX 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 AIX 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 コネクター始動スクリプト内のクラスパスを変更してください。

Windows ファイル構造

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

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

表 8 に、コネクターが使用する Windows ファイル構造の説明と、インストーラーからコネクター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。

表 8. Windows ファイル構造

***ProductDir* のサブディレクトリー**	**説明**
connectors¥WebSphereMQ¥CWWebSphereMQ.jar	WebSphere MQ コネクターによってのみ使用されるクラスを含む
connectors¥WebSphereMQ¥start_WebSphereMQ.bat	コネクター (NT/2000) の始動スクリプト
connectors¥messages¥WebSphereMQConnector.txt	コネクターのメッセージ・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
connectors¥WebSphereMQ¥samples¥LegacyContact¥WebSphereMQConnector.cfg	WebSphere MQ 構成ファイルのサンプル
connectors¥WebSphereMQ¥samples¥LegacyContact¥PortConnector.cfg	ポート・コネクター構成ファイルのサンプル
connectors¥WebSphereMQ¥samples¥LegacyContact¥Sample_WebSphereMQ_LegacyContact.xsd	スキーマのサンプル
connectors¥WebSphereMQ¥samples¥LegacyContact¥Sample_WebSphereMQ_MO_Config.xsd	メタオブジェクトのサンプル

表 8. Windows ファイル構造 (続き)

***ProductDir* のサブディレクトリー**	**説明**
connectors¥WebSphereMQ¥samples¥LegacyContact¥Sample_WebSphereMQ_MO_DataHandler.xsd	データ・ハンドラー・メタオブジェクトのサンプル
connectors¥WebSphereMQ¥samples¥LegacyContact¥Sample_WebSphereMQ_MO_DataHandler_DelimitedConfig.xsd	区切りデータ・ハンドラー・メタオブジェクトのサンプル
connectors¥WebSphereMQ¥samples¥LegacyContact¥Sample_WebSphereMQ_DynMO_Config.xsd	動的なメタオブジェクトのサンプル
connectors¥WebSphereMQ¥samples¥LegacyContact¥JMSPropertyPairs.xsd	動的なメタオブジェクトの子ビジネス・オブジェクトの JMS プロパティーのサンプル

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。

UNIX のファイル構造

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

ユーティリティーにより、コネクターが *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、コネクターが使用する UNIX ファイル構造の説明と、インストーラーからコネクター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。

***ProductDir* のサブディレクトリー**	**説明**
connectors/WebSphereMQ/CWWebSphere MQ.jar	WebSphere MQ コネクターによってのみ使用されるクラスを含む
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。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。
connectors/messages/WebSphereMQ/Connector.txt	コネクターのメッセージ・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
connectors/WebSphereMQ/samples/LegacyContact/WebSphereMQConnector.cfg	WebSphere MQ 構成ファイルのサンプル
connectors/WebSphereMQ/samples/LegacyContact/PortConnector.cfg	ポート・コネクター構成ファイルのサンプル

ProductDir のサブディレクトリー	説明
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _LegacyContact.xsd	サンプル・ビジネス・オブジェクト・リポジトリー・ファイル
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _MO_Config.xsd	メタオブジェクトのサンプル
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _MO_DataHandler.xsd	データ・ハンドラー・メタオブジェクトのサンプル
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _MO_DataHandler_DelimitedConfig.xsd	区切りデータ・ハンドラー・メタオブジェクトのサンプル
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _DynMO_Config.xsd	動的なメタオブジェクトのサンプル
connectors/WebSphereMQ/samples/LegacyContact/JMSPropertyPairs.xsd	動的なメタオブジェクトの子ビジネス・オブジェクトの JMS プロパティーのサンプル

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。

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 を使用します。

- Connector Configurator の説明と段階的な手順については、87 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、『標準コネクター・プロパティー』、および 59 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。
- コネクター固有のプロパティーの詳細については、25 ページの『コネクター固有のプロパティー』を参照してください。

コネクターは、始動時に構成値を取得します。実行時セッション中に、1 つ以上のコネクター・プロパティーの値の変更が必要になることがあります。AgentTraceLevel など一部のコネクター構成プロパティーへの変更は、即時に有効になります。その他のコネクター・プロパティーへの変更を有効にするには、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。あるプロパティーが動的 (即時に有効になる) か静的 (コネクター・コンポーネントまたはシステムを再始動する必要がある) かを判別するには、Connector Configurator の「コネクター・プロパティー」ウィンドウ内の「更新メソッド」列を参照してください。

標準コネクター・プロパティー

標準構成プロパティーにより、すべてのコネクターによって使用される情報が提供されます。標準構成プロパティーの資料については、59 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

注: Connector Configurator で構成プロパティーを設定するときは、BrokerType プロパティーで使用するブローカーを指定します。このプロパティーの値を設定すると、使用するブローカーに関連するプロパティーが「Connector Configurator」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ー

コネクター固有の構成プロパティーには、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ーは、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の構成プロパティーを示します。プロパティーの説明については、以下の各セクションを参照してください。

名前	指定可能な値	デフォルト値	必須
ApplicationPassword	*ログイン・パスワード*		いいえ
ApplicationUserName	*ログイン・ユーザー ID*		いいえ
ArchiveQueue	*正常に処理されたメッセージのコピー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
DataHandlerPoolSize	*再利用のためにキャッシュするデータ・ハンドラー・インスタンスの数*	30	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
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	*InputQueue プロパティー 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	*同期要求処理時の受信側のポーリング間隔 (ミリ秒単位)*		いいえ
SecurityExitClassName	*使用されるセキュリティー出口の完全修飾クラス名*		いいえ

名前	指定可能な値	デフォルト値	必須
SecurityExitInitParam	*セキュア出口を呼び出すために使用される初期化ストリングによって構成する値を指定します。*		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
WorkerThreadCount	*ポーリングのための並列スレッドの最大数*	1	いいえ

ApplicationPassword

WebSphere MQ にログインするために、UserID と共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 と共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) の設定』を参照してください。

デフォルト = 設定値なし

Channel

コネクターが WebSphere MQ と通信するときに使用する MQ サーバー・コネクター・チャネルです。

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、コネクターは WebSphere MQ によって提供されるデフォルトのサーバー・チャネルを使用します。注 *

ConfigurationMetaObject

コネクター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

DataHandlerPoolSize

特定のタイプのデータ・ハンドラー用にキャッシュするデータ・ハンドラー・インスタンスの最大数を指定できます。

デフォルト = 30

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 プロパティー。

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 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ーです。このプロパティーを使用すると、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 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。コネクターは、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

コネクターは、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

表 9 に、サンプル・メタオブジェクトを示します。

表 9. メタオブジェクトのサンプル

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ー *InputQueue* の値を MyQueueA;MyQueueB;MyQueueC とします。

InputQueue プロパティー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、MqQueueB 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 名をコネクター固有のプロパティーから除去

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ーを構成できます。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ー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場合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 プロパティー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、45 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

同期要求処理時の受信側のポーリング間隔を指定します。値はミリ秒単位です。

デフォルト = 設定値なし

SecurityExitClassName

使用されるセキュリティー出口の完全修飾クラス名。

デフォルト = 設定値なし

SecurityExitInitParam

セキュア出口を呼び出すために使用される初期化ストリングによって構成する値を指定します。

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、コネクターは、各 is Required 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。このパラメーターを `false` に設定すると、コネクター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。

WorkerThreadCount

ポーリングのための並列スレッドの最大数。並行してイベントを処理している間、アダプターはイベントを受信した順序でブローカーに実行依頼することができなくなります。順序を維持する必要がある場合は、WorkerThreadCount を常に 1 に設定する必要があります。

複数コネクター・インスタンスの作成

コネクターの複数インスタンスの作成は、多くの点でカスタム・コネクター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、コネクターの複数のインスタンスを作成して実行するように、ご使用のシステムを設定することができます。それには、以下の作業を行う必要があります。

- コネクター・インスタンスの新規ディレクトリーを作成する
- 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
- 新規コネクター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

新規ディレクトリーの作成

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

```
ProductDir¥connectors¥connectorInstance
```

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

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

```
ProductDir¥repository¥connectorInstance
```

ビジネス・オブジェクト定義の作成

プロジェクト内にコネクター・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. 初期コネクターに関連付けられているビジネス・オブジェクト定義を変更する必要がある場合は、適切なファイルをコピーし、Business Object Designer を使用してそれらのファイルをインポートします。初期コネクター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. 初期コネクターのファイルは、次のディレクトリーに入っていなければなりません。

   ```
   ProductDir¥repository¥initialConnectorInstance
   ```

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

コネクター定義の作成

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

1. 初期コネクターの構成ファイル (コネクター定義) をコピーし、ファイルを名前変更する。
2. 各コネクター・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. 適宜コネクター・プロパティー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. 初期コネクターの始動スクリプトをコピーし、そのファイルに次のコネクター・ディレクトリー名が含まれる名前を付ける。

   dirname
2. この始動スクリプトを 31 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに置く。
3. 始動スクリプトのショートカットを作成する (Windows のみ)。
4. 初期コネクターのショートカット・テキストをコピーして、コマンド行から新規コネクター・インスタンスに合うように初期コネクター名を変更する。

これにより、Integration Server で両方のコネクター・インスタンスを同時に実行できます。

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

キューの Uniform Resource Identifier (URI) の設定

URI を使用して、キューを指定し、それらのキューの値を設定または変更することができます。キューを定義するコネクター固有のプロパティーに値を指定するときに実行します。

キュー URI を設定するには、以下の手順を実行します。

Connector Configurator を使用して、以下の構文に従ってコネクター固有のキューを指定します。

キュー URI 構文: 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・プロパティーの、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
```

表 10 に、キュー URI のプロパティー名を示します。

表 10. キュー URI のプロパティー名

プロパティー名	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、このプロパティー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、コネクター自身のデフォルト値を使用できるよう指定します。
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、このプロパティー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、コネクター自身のデフォルト値を使用できるよう指定します。
CCSID	アウトバウンド・メッセージの文字セット・エンコード	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 コネクター固有の構成プロパティー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・バッファーから

検索されるかメッセージ・バッファーにデリバリーされるときに行われるため、コネクターは JMS の IBM WebSphere MQ インプリメンテーションに依存してデータ変換を行います (IBM WebSphere MQ Java クライアント・ライブラリー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。コネクターは、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。コネクターは、特別な変更を必要とせずに、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。コネクターはこの情報を WebSphere MQ に渡し、WebSphere MQ は MQMessage のデリバリーのためにデータをエンコードするときに (JMS API を介して) この情報を使用します。CCSID およびエンコードがサポートされていなくても、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、多くの場合解決できます。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、IBM ソフトウェア・サポートに連絡し、代替の Java 仮想マシンを使用してコネクター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

コネクターは、メタオブジェクト・エントリーを使用して、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。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 メッセージの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、コネクターは、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・オブジェクト名と共に、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、コネクターはそのビジネス・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、gotApplEvents() 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。

コネクターは、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。

- 静的なコネクター・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・プロパティー

表 11 に、メタオブジェクトでサポートされるプロパティー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ーの説明を参照してください。メタオブジェクトには、表 11 に示すプロパティーが 1 つ以上含まれていなければなりません。

一部のプロパティーは、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ーもあります。特定のプロパティーについて、コネクター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 11. WebSphere MQ アダプター・メタオブジェクトのプロパティー

プロパティー名	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	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。 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、 40 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。 この条件が満たされていない場合は、コネクターが 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とランタイム・エラーが発生します。 **注:** このプロパティーは、同期要求にのみ利用可能です。
DataHandlerConfigMO	はい	はい	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 コネクター・プロパティー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・プロパティーは、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、コネクター・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。 87 ページの『付録 B. Connector Configurator』の説明を参照してください。

表 11. WebSphere MQ アダプター・メタオブジェクトのプロパティー (続き)

プロパティー名	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 コネクター・プロパティー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・プロパティーは、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、このプロパティーの値に対応する属性を含める必要があります。 87 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	87 ページの『付録 B. Connector Configurator』の説明を参照してください。
InputFormat	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、コネクターは特定のビジネス・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。このプロパティーを設定するときは、デフォルトのメタオブジェクト変換プロパティー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ーの値は、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。コネクター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、コネクター固有のプロパティー MessageFormatProperty を使用してユーザーが定義できます。
OutputFormat	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 の名前の長さは、16 文字を超えてはなりません。超えた場合、WebSphere MQ がエラーを生成する場合があります。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。

表 11. WebSphere MQ アダプター・メタオブジェクトのプロパティー (続き)

プロパティー名	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	コネクターが、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。このプロパティーは、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。このプロパティーを設定するときは、デフォルトの変換プロパティーを使用しないでください。デフォルトの変換プロパティーの値は、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。 **注:** コネクター固有のプロパティーにある InputQueue プロパティーは、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ーです。静的 MO では、InputQueue プロパティーおよび InputFormat プロパティーは、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、コネクター固有のプロパティー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、42 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。
OutputQueue	はい	はい	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。
ResponseTimeout	はい	はい	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。このプロパティーが未定義のままか、またはゼロよりも小さい値に設定されている場合、コネクターは応答を待機せず、SUCCESS を即時に戻します。
TimeoutFatal	はい	はい	同期要求処理で、応答の受信がないためコネクターからエラー・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。このプロパティーの値が True の場合、コネクターは、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。このプロパティーが未定義の場合、または False に設定されている場合、コネクター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。

表 11. WebSphere MQ アダプター・メタオブジェクトのプロパティー (続き)

プロパティー名	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。このプロパティーが静的メタオブジェクトで指定されていない場合、コネクター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。動的子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。可能な値は、text、binary、または object です。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。すなわち、Text:ISO8859_1、Text:UnicodeLittle、Text、Binary、または Object のようになります。このプロパティーは内部的に InputFormat プロパティーに関連します。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、受信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。

表 11. WebSphere MQ アダプター・メタオブジェクトのプロパティー (続き)

プロパティー名	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
JMSProperties		はい	45 ページの『JMS プロパティー』を参照してください。

バイナリーおよびオブジェクト・メッセージ用の DataEncoding

メタオブジェクト・プロパティーである DataEncoding を使用して、メッセージ・タイプを変更することができます。このプロパティーは、text、binary、object という 3 つの値のいずれかを受け入れます。

デフォルトでは、アダプターは、すべてのメッセージのタイプが text であることを前提とします。アダプターは、バイナリー・メッセージを受信した場合、内容を構成済みデータ・ハンドラーに渡す前に、Java 仮想マシン (JVM) のデフォルト・エンコード方式を使用してバイナリーの内容をテキストに変換します。メタオブジェクトの DataEncoding プロパティー内の binary または object メッセージ・タイプを明示すると、この振る舞いは変化します。

- メタオブジェクト内の DataEncoding プロパティーを使用して binary を指定した場合、アダプターは以下のように動作します。
  1. 要求の処理時に、アダプターはビジネス・オブジェクトをデータ・ハンドラーのバイナリー・メソッドに渡し、バイト・メッセージをデリバリーします。
  2. イベント通知時に、アダプターはバイナリー・メッセージからバイトを検索し、Java InputStream インスタンス (バイト) としてデータ・ハンドラーに渡します。
  3. テキスト・メッセージを受信した場合、アダプターは、内容をデータ・ハンドラーに渡す前に、JVM のデフォルト・エンコード方式を使用して、テキスト本文をバイナリーに変換します。
- メタオブジェクト内の DataEncoding プロパティーを使用して object を指定した場合、アダプターは以下のように動作します。
  1. 要求の処理時に、アダプターはビジネス・オブジェクトをデータ・ハンドラーの getStreamFromBO() メソッドに渡し、ObjectInputStream を取得します。
  2. イベント通知時に、アダプターはオブジェクト・メッセージから Java オブジェクトを検索し、Java ObjectInputStream インスタンス (バイト) としてデータ・ハンドラーに渡します。

  注: すべてのデータ・ハンドラーが binary および object データをサポートするわけではありません。構成済みのデータ・ハンドラーでサポートされているかどうかを必ずチェックしてください。

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ー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 を起動し、アダプターに付属しているサンプル connectors¥WebSpereMQ¥samples¥LegacyContact¥WebSpereMQ_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・プロパティー 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共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・プロパティー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 12. 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 13. 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) のプロパティーを 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ーと、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ーが結合されて、メタオブジェクトの属性の最終的なプロパティーになります。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ーが 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、コネクターは、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・プロパティーに加えて OutputDestination=QueueA が指定されていると見なします。

表 14. 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
Default	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ーは、36 ページの表 11 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、コネクター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、コネクターはビジネス・オブジェクトを指定せずに、メッセージ・テキストを構成済みのデータ・ハンドラーに渡します。データ・ハンドラーがテキスト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、コネクターはこのメッセージ・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 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 (36 ページの表 11 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、コネクター・プロパティー 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 プロパティー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. コネクター固有のプロパティー (29 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** 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、変換プロパティー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ーと重複します。

動的な子メタオブジェクトのプロパティーは静的なメタオブジェクトから検出されるプロパティー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・プロパティー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ーは、36 ページの表 11 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・プロパティーを反映しており、動的メタオブジェクトで認識されます。

表 15. 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
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

表 15. 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
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、コネクターは、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 16 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 16. 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
InputQueue	MYInputQueue
OutputFormat	CxIgnore
OutputQueue	CxIgnore
ResponseTimeout	CxIgnore
TimeoutFatal	CxIgnore

表 16 に示すように、動的子メタオブジェクトに、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ー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ー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 プロパティー: 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ー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ー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 プロパティー・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・プロパティー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

46 ページの表 17 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ーを示します。

表 17. JMS プロパティー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 プロパティー名 (有効とは、プロパティーの型と ASI で定義した型が矛盾しないこと)	name=<*JMS プロパティー名*>;type=<*JMS プロパティーの型*>	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ー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・プロパティー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 プロパティーのタイプです。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 プロパティーのタイプによって、どのメソッドを使用してメッセージのプロパティー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 つのプロパティー (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 プロパティー ID にマップされます。

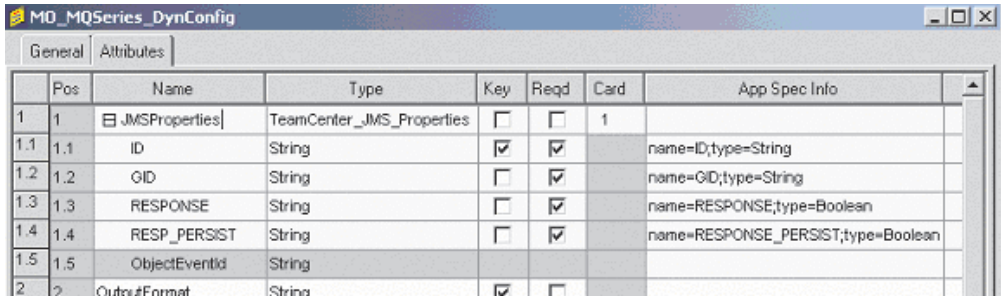

図 3. 動的メタオブジェクトの JMS プロパティー属性

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** 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
```

   コネクターは、上の定義にあてはまる要求を受信すると、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

WebSphere MQ 用コネクターを始動するためには、始動ファイルの構成が必要です。

Windows

Windows プラットフォーム用のコネクターの構成を完成させるには、start_WebSphereMQ.bat ファイルを変更する必要があります。

1. start_WebSphereMQ.bat ファイルを開きます。
2. スクロールして「Set the directory containing your WebSphere MQ Java client libraries」で始まるセクションに移動し、WebSphere MQ Java クライアント・ライブラリーの場所を指定します。

UNIX

UNIX プラットフォーム用のコネクターの構成を完成させるには、start_WebSphereMQ.sh ファイルを変更する必要があります。

1. start_WebSphereMQ.sh ファイルを開きます。
2. スクロールして「Set the directory containing your WebSphere MQ Java client libraries」で始まるセクションに移動し、WebSphere MQ Java クライアント・ライブラリーの場所を指定します。

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 システムの場合は、始動スクリプトは、次に示すようなコネクターのランタイム・ディレクトリーに存在している必要があります。

ProductDir¥connectors¥*connName*

このディレクトリー名の *connName* はコネクターを表しています。

UNIX システムの場合は、始動スクリプトは、*UNIX ProductDir*/bin ディレクトリーに存在している必要があります。

表 18 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 18. コネクターの始動スクリプト

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

始動スクリプトは、実行時に、デフォルトでは *Productdir* に構成ファイルがあることを前提とします (下記のコマンドを参照してください)。ここに、構成ファイルを置いてください。

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

以下のいずれかの方法で、コネクター始動スクリプトを起動できます。

- Windows システムでは、「**スタート**」 メニューから、

「プログラム」>「IBM WebSphere Business Integration Adapters」>「Adapters」>「Connectors」を選択します。デフォルトでは、プログラム名は「IBM WebSphere Business Integration Adapters」ですが、カスタマイズもできます。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。

- コマンド行から起動する。
  - Windows システム:

    ```
    start_connName connName brokerName [-cconfigFile ]
    ```

  - UNIX 系システム:

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

    ここで、*connName* はコネクター名であり、*brokerName* はご使用の統合ブローカーを表しています。以下に例を示します。

  - WebSphere InterChange Server では、*brokerName* に ICS インスタンスの名前を指定する
  - WebSphere Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、または WebSphere Business Integration Message Broker)、または WebSphere Application Server では、*brokerName* にブローカーを識別するストリングを指定する

    注: Windows システム上の WebSphere Message Broker または WebSphere Application Server では、-c オプションの後にコネクター構成ファイルの名前を*指定する必要があります*。ICS では -c はオプションです。

- WebSphere Application Server または InterChange Server ブローカーと共に稼動する System Manager を始動したときに起動される Adapter Monitor から起動する。

  このツールを使用して、コネクターをロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除できます。

- System Manager (すべてのブローカーで使用可能) から起動する。

  このツールを使用して、コネクターをロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除できます。

- Windows システムでは、コネクターを Windows のサービスとして開始す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。

コマンド行始動オプションを含むコネクターの開始方法についての詳細は、以下の資料を参照してください。

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

コネクターの停止

コネクターの停止方法は、以下のように、コネクターを始動した方法により異なります。

- コネクターの始動スクリプトを使用してコネクターをコマンド行から始動した場合:
  - Windows システムでは、始動スクリプトを起動すると、そのコネクター用の個別の「コンソール」ウィンドウが作成されます。このウィンドウで、「Q」と入力して Enter キーを押すと、コネクターが停止します。
  - UNIX 系システムで InterChange Server を使用する場合は、コネクターはバックグラウンドで稼働するため、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ーを停止します。

    ```
    connector_manager_connName -stop
    ```

    ここで、*connName* はコネクター名です。
- Adapter Monitor から起動する (WebSphere Business Integration Adapters 製品のみ)。System Manager を始動すると起動されます。

  このツールを使用して、コネクターをロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除できます。
- System Monitor から起動する (WebSphere InterChange Server 製品のみ)。

  このツールを使用して、コネクターをロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除できます。
- Windows システムでは、コネクターを Windows のサービスとして開始するように構成できる。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ーは停止します。

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』
- 54 ページの『エラー処理』
- 55 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ーの設定、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に対して定められている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 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

ーのデータ・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ん。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・プロパティーのサンプル

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・プロパティー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

コネクター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、標準コネクター構成プロパティー 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
```

コネクターは、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「APPRESPONSETIMEOUT」は、以下の場合に戻されます。

- コネクターは、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。
- コネクターはビジネス・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。
- コネクターはメッセージを発行しましたが、変換プロパティー 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。
- コネクターは、戻りコードが APPRESPONSETIMEOUT または UNABLETOLOGIN 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、コネクターは UnsubscribedQueue プロパティー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。

gotApplEvent() メソッドによって NO_SUBSCRIPTION_FOUND コードが戻されると、コネクターは UnsubscribedQueue プロパティー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。

コネクターがアクティブでない

gotApplEvent() メソッドが CONNECTOR_NOT_ACTIVE コードを戻すと、pollForEvents() メソッドは APPRESPONSETIMEOUT 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・ハンドラーによる変換

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、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ー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』のコネクター構成プロパティーの説明を参照してください。トレースの使用可能化や設定方法などの詳細については、「*コネクター開発ガイド*」を参照してください。

コネクターのトレース・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　コネクター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　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　ビジネス・オブジェクトが gotApplEvent() または executeCollaboration() から InterChange Server 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　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　コネクター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　コネクター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、アダプターがキャッチした例外で 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	考えられる処置/説明
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...`	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。
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...`	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。
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...`	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。
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` [Solaris]) を見つけることができません。パスに IBM WebSphere MQ Java クライアント・ライブラリーのフォルダーが含まれていることを確認します。
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 ':'`	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	考えられる処置/説明
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。	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	考えられる処置/説明
コネクターは、コネクター・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。	これは WebSphere MQ MQMD メッセージ・ヘッダーの制限であり、コネクターの制限ではありません。

付録 A. コネクターの標準構成プロパティー

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

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

ご使用のアダプターが DB2 Information Integrator をサポートする場合、WMQI オプションおよび DB2 II 標準プロパティーを使用します (61 ページの表 19 の注の欄を参照してください)。

アダプター用に設定するプロパティーは、使用する統合ブローカーによって異なります。Connector Configurator を使用して統合ブローカーを選択します。ブローカーを選択すると、Connector Configurator では、アダプター用に構成する必要のある標準プロパティーがリストされます。

このコネクター固有のプロパティーについては、本書の関連セクションを参照してください。

新規プロパティー

この標準プロパティーは、今回のリリースで追加されました。

- BOTrace

標準コネクター・プロパティーの概要

コネクターには 2 つのタイプの構成プロパティーがあります。

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

これらのプロパティーは、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 の始動方法を説明し、すべてのプロパティーに共通する特性を説明します。コネクター固有の構成プロパティーについては、該当するアダプターのユーザーズ・ガイドを参照してください。

Connector Configurator の始動

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

Connector Configurator と System Manager は、Windows システム上でのみ動作します。コネクターを UNIX システム上で稼働している場合でも、これらのツールがインストールされた Windows マシンが必要です。

UNIX 上で動作するコネクターのコネクター・プロパティーを設定する場合は、Windows マシン上で System Manager を起動し、UNIX の統合ブローカーに接続してから、コネクター用の Connector Configurator を開く必要があります。

構成プロパティー値の概要

コネクターは、以下の順序に従ってプロパティーの値を決定します。

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

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

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

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

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

- **動的**
  変更を System Manager に保管すると、新規の値が即時に有効になります。コネクターがスタンドアロン・モードで (System Manager から独立して) 作動している場合 (例えば、WebSphere Message Brokers の 1 つで作動している場合)、構成ファイルによってのみプロパティーの変更が可能です。この場合、動的更新は実行できません。
- **エージェント再始動 (ICS のみ)**
  コネクター・エージェントを停止して再始動しなければ、新規の値が有効になりません。
- **コンポーネント再始動**
  System Manager でコネクターを停止してから再始動しなければ、新規の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、再始動する必要はありません。

- **システム再始動**
  コネクター・エージェントおよびサーバーを停止して再始動しなければ、新規の値が有効になりません。

特定のプロパティーの更新方法を確認するには、「Connector Configurator」ウィンドウ内の「**更新メソッド**」列を参照するか、61 ページの表 19 の「更新メソッド」列を参照してください。

標準プロパティーが存在する場所は 3 つあります。一部のプロパティーは複数の場所にあってもかまいません。

- **ReposController**
  このプロパティーはコネクター・コントローラー内にあり、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、コントローラーには影響しません。
- **ReposAgent**
  このプロパティーはエージェント内にあり、その場所でのみ有効です。プロパティーによっては、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**
  このプロパティーは、コネクター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。コントローラーはこのプロパティー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ー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。

標準プロパティーの早見表

表 19 は、標準コネクター構成プロパティーの早見表です。すべてのコネクターでこれらのプロパティーすべてを必要とするわけではなく、プロパティーの設定は統合ブローカーごとに異なる場合があります。

各プロパティーの説明については、表の次のセクションを参照してください。

注: 表 19 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 であることを示します。ブローカーが WMQI または WAS の場合には、リポジトリー・ディレクトリーは <*ProductDir*>¥repository に設定されます。

表 19. 標準構成プロパティーの要約

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
AdapterHelpName	有効な <*RegionalSetting*> 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 1 つ	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、and enu_usa (デフォルト) を含む。

表 19. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
AdminInQueue	有効な JMS キュー名	*<CONNECTORNAME>* /ADMININQUEUE	コンポーネント 再始動	このプロパティーは、 DeliveryTransport の 値が JMS 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	このプロパティーは、 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	このプロパティーは、 DeliveryTransport の 値が MQ または IDL であり、 RepositoryDirectory の値が <REMOTE> に 設定され、BrokerType の 値が ICS である場合のみ 有効です。
AgentTraceLevel	0 から 5	0	ブローカーが ICS の場合は 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・ アプリケーション名 として指定された値	コンポーネント 再始動	
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	このプロパティーは、 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	このプロパティーは、 BiDi.Transformation の値が true の 場合のみ有効です。 BrokerType の値が ICS の場合、 プロパティーは 読み取り専用です。

表 19. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
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	このプロパティーは、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	このプロパティーは、BrokerType の値が WAS でない場合のみ有効です。
BOTrace	none または keys または full	none	エージェント再始動	このプロパティーは、AgentTraceLevel の値が 5 より小さい場合のみ有効です。
BrokerType	ICS、WMQI、WAS	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。 次のリストはそのサブセットです。 ascii7、ascii8、SJIS、Cp949、GBK、Big5、Cp297、Cp273、Cp280、Cp284、Cp037、Cp437	ascii7	コンポーネント再始動	このプロパティーは、C++ コネクターでのみ有効です。
CommonEventInfrastructure	true または false	false	コンポーネント再始動	
CommonEventInfrastructureURL	URL ストリング。例えば、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	このプロパティーは、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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
ContainerManagedEvents	ブランクまたは JMS	ブランク	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
ControllerEventSequencing	true または false	true	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。

表 19. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
ControllerStoreAndForwardMode	true または false	true	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
ControllerTraceLevel	0 から 5	0	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
DeliveryQueue	任意の有効な JMS キュー名	*<CONNECTORNAME>*/DELIVERYQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
DeliveryTransport	MQ、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、このプロパティーの有効な値は JMS のみです。
DuplicateEventElimination	true または false	false	コンポーネント再始動	このプロパティーは、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	このプロパティーは、BrokerType の値が ICS である場合のみ有効です。
FaultQueue	任意の有効なキュー名	*<CONNECTORNAME>*/FAULTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS 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	このプロパティーは、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	このプロパティーは、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	crossworlds.queue.manager	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。

表 19. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
jms.Password	任意の有効なパスワード		コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS 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	このプロパティーは、DeliveryTransport の値が JMS であり、BrokerType の値が ICS である場合のみ有効です。
jms.UserName	任意の有効な名前		コンポーネント再始動	このプロパティーは、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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
ListenerConcurrency	1 から 100	1	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が MQ 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サブセットです。 en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR	en_US	コンポーネント再始動	
LogAtInterchangeEnd	true または false	false	コンポーネント再始動	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。

表 19. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
MaxEventCapacity	1 から 2147483647	2147483647	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
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	このプロパティーは、DuplicateEventElimination の値が true であり、ContainerManagedEvents が値を持たない場合のみ有効です。
OADAutoRestartAgent	true または false	false	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
OADMaxNumRetry	正整数	1000	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
OADRetryTimeInterval	正整数 (単位: 分)	10	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
PollEndTime	HH = 0 から 23 MM = 0 から 59	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒)	10000	ブローカーが ICS の場合は動的、その他の場合はコンポーネント再始動	
PollQuantity	1 から 500	1	エージェント再始動	このプロパティーは、ContainerManagedEvents の値が JMS 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	

表 19. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
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。 WMQI および WAS の場合、値は *<ProductDir>* ¥repository です。	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	このプロパティーは、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	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
RestartRetryCount	0 から 99	7	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 の値 (単位: 分)	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	
ResultsSetEnabled	true または false	false	コンポーネント再始動	DB2II をサポートするコネクターのみが使用します。 このプロパティーは、DeliveryTransport の値が JMS であり、BrokerType の値が WMQI である場合のみ有効です。
ResultsSetSize	正整数	0 (結果セット・サイズが無制限であることを意味する)	コンポーネント再始動	DB2II をサポートするコネクターのみが使用します。 このプロパティーは、ResultsSetEnabled の値が true の場合のみ有効です。
RHF2MessageDomain	mrm または xml	mrm	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS であり、WireFormat の値が CwXML である場合のみ有効です。
SourceQueue	任意の有効な WebSphere MQ キュー名	*<CONNECTORNAME>* /SOURCEQUEUE	エージェント再始動	このプロパティーは、ContainerManagedEvents の値が JMS の場合のみ有効です。

表 19. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
SynchronousRequest Queue	任意の有効なキュー名	*<CONNECTORNAME>* /SYNCHRONOUSREQUEST QUEUE	コンポーネント 再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
SynchronousRequest Timeout	0 から任意の数 (ミリ秒)	0	コンポーネント 再始動	このプロパティーは、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	このプロパティーは、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、このプロパティーの値は、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	このプロパティーは、BrokerType の値が WAS の場合のみ有効です。
XMLNameSpaceFormat	short または long または no	short	エージェント 再始動	このプロパティーは、BrokerType の値が WMQI または WAS である場合のみ有効です。

標準プロパティー

このセクションでは、標準コネクター構成プロパティーについて説明します。

AdapterHelpName

AdapterHelpName プロパティーは、コネクター固有の全般ヘルプ・ファイルがあるディレクトリーの名前です。ディレクトリーは、*<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ー 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ーが含まれることがあります。

デフォルト値は、有効な場合はテンプレート名、そうでなければブランクです。

AdminInQueue

AdminInQueue プロパティーは、統合ブローカーからコネクターへ管理メッセージが送信されるときに使用されるキューを指定します。

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

AdminOutQueue

AdminOutQueue プロパティーは、コネクターから統合ブローカーへ管理メッセージが送信されるときに使用されるキューを指定します。

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

AgentConnections

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

RepositoryDirectory の値が <REMOTE> に設定され、DeliveryTransport プロパティーの値が MQ または IDL である場合のみ有効です。

このプロパティーのデフォルト値は 1 です。

AgentTraceLevel

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

デフォルト値は 0 です。

ApplicationName

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

デフォルトはコネクターの名前です。

BiDi.Application

BiDi.Application プロパティーは、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、双方向形式を指定します。このプロパティーは、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)
- シェーピング (アラビア語): オンまたはオフ (S または N)

- 数字シェーピング (アラビア語): ヒンディ語、コンテキスト、または公称 (H、C、または N)

このプロパティーは、BiDi.Transformation プロパティーの値が 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、公称) です。

BiDi.Broker

BiDi.Broker プロパティーは、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、双方向スクリプト形式を指定します。データの双方向属性を定義します。属性は、前述の BiDi.Application の説明でリストされています。

このプロパティーは、BiDi.Transformation プロパティーの値が true に設定されている場合のみ有効です。BrokerType プロパティーが ICS の場合、プロパティー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、公称) です。

BiDi.Metadata

BiDi.Metadata プロパティーは、メタデータの双方向形式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、詳しくはアダプター固有のプロパティーに関するセクションを参照してください。

このプロパティーは、BiDi.Transformation プロパティーの値が 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、公称) です。

BiDi.Transformation

BiDi.Transformation プロパティーは、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。

プロパティー値が true に設定されている場合、BiDi.Application、BiDi.Broker、および BiDi.Metadata プロパティーが使用可能です。プロパティー値が 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。

BOTrace

BOTrace プロパティーは、実行時にビジネス・オブジェクト・トレース・メッセージを使用可能にするかどうかを指定します。

注: AgentTraceLevel プロパティーが 5 より小さい値に設定された場合にのみ、適用されます。

トレース・レベルが 5 よりも小さい値に設定された場合は、以下のコマンド行パラメーターを使用して、BOTrace の値をリセットすることができます。

- ビジネス・オブジェクトのすべての属性をダンプするには、-xBOTrace=Full を入力します。
- ビジネス・オブジェクトのキーのみをダンプするには、-xBOTrace=Keys を入力します。
- ビジネス・オブジェクト属性のダンプを使用不可にするには、-xBOTrace=None を入力します。

デフォルト値は false です。

BrokerType

BrokerType プロパティーは、使用している統合ブローカーのタイプを識別します。指定可能な値は ICS、WMQI (WMQI、WMQIB、または WBIMB の場合)、または WAS です。

CharacterEncoding

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

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

デフォルトでは、サポートされる文字エンコードのサブセット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。詳しくは、本書の付録『Connector Configurator』を参照してください。

CommonEventInfrastructure

Common Event Infrastructure (CEI) は、生成されたイベントを処理する単純なイベント管理機能です。CommonEventInfrastructure プロパティーは、CEI 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。

CommonEventInfrastructureContextURL

CommonEventInfrastructureContextURL は、Common Event Infrastructure (CEI) サーバー・アプリケーションを実行する WAS サーバーへのアクセスを獲得するために使用されます。このプロパティーは、使用される URL を指定します。

このプロパティーは、CommonEventInfrastructure の値が true に設定されている場合のみ有効です。

デフォルト値はブランクのフィールドです。

ConcurrentEventTriggeredFlows

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

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

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

- Maximum number of concurrent events プロパティーの値を十分に大きい値に設定して、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実行できるように構成する必要があります。つまり、このコンポーネントがマルチスレッド化されているか、またはコネクター・エージェント並列処理を使用でき、複数プロセスに対応するよう構成されている必要があります。Parallel Process Degree 構成プロパティーに、1 より大きい値を設定する必要があります。

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

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

このプロパティーを JMS に設定した場合には、保証付きイベント・デリバリーを使用できるように次のプロパティーも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) プロパティーを設定したデータ・ハンドラーも構成する必要があります。DataHandlerConfigMOName (オプションのメタオブジェクト名) を追加することもできます。これらのプロパティーの値を設定するには、Connector Configurator の「**データ・ハンドラー**」タブを使用します。

これらのプロパティーはアダプター固有ですが、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 プロパティーを 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 プロパティーは、DeliveryTransport 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 プロパティーは、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。

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

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

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

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

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

デフォルト値は true です。

ControllerTraceLevel

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

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

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

- RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合、DeliveryTransport プロパティーには MQ、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 プロパティーの値がローカル・ディレクトリーの場合、値に使用できるのは JMS のみです。

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

DeliveryTransport プロパティーの値が MQ である場合、アダプター始動スクリプト内のコマンド行パラメーター WhenServerAbsent を設定して、InterChange Server のシャットダウン時にアダプターが一時停止するかシャットダウンするかを指示することができます。

- ICS が使用不可になったときにアダプターを一時停止するには、WhenServerAbsent=pause と入力します。

- ICS が使用不可になったときにアダプターをシャットダウンするには、`WhenServerAbsent=shutdown` と入力します。

WebSphere MQ および IDL

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

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

JMS

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

JMS をデリバリー・トランスポートとして選択した場合は、`jms.MessageBrokerName`、`jms.FactoryClassName`、`jms.Password`、および `jms.UserName` などの追加の JMS プロパティーが Connector Configurator 内にリストされます。プロパティー `jms.MessageBrokerName` および `jms.FactoryClassName` は、このトランスポートの必須プロパティーです。

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

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

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

- CWSharedEnv.sh スクリプト内で 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・エディターを使用して、CWSharedEnv.sh スクリプトの最初の行として次の行を追加します。

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

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

DuplicateEventElimination

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

注: このプロパティーの値が 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 プロパティー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

このプロパティーの値が 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。

このプロパティーは、BrokerType プロパティーが 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 (および状況表示と問題説明) を FaultQueue プロパティー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 プロパティーは、JMS プロバイダーのためにインスタンスを生成するクラス名を指定します。DeliveryTransport プロパティーの値が JMS に設定されている場合、このプロパティーを設定する必要があります。

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

jms.ListenerConcurrency

jms.ListenerConcurrency プロパティーは、JMS コントローラーの並行リスナーの数を指定します。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。

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

デフォルト値は `1` です。

jms.MessageBrokerName

jms.MessageBrokerName 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 プロパティーで) 指定する場合、このコネクター・プロパティーを設定する必要があります。

リモート・メッセージ・ブローカーに接続した場合、このプロパティーでは以下の値を指定する必要があります。
QueueMgrName:*Channel*:*HostName*:*PortNumber*
各変数の意味は、以下のとおりです。
QueueMgrName は、キュー・マネージャー名です。
Channel は、クライアントが使用するチャネルです。
HostName は、キュー・マネージャーの配置先のマシン名です。
PortNumber は、キュー・マネージャーが listen に使用するポートの番号です。

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 プロパティーは、コネクターに対して同時に送信することができる並行サービス呼び出し要求の数 (最大値) を指定します。この最大値に達した場合、新規のサービス呼び出し要求はブロックされ、処理を続行するには既存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 プロパティーは、JMS プロバイダーのためのパスワードを指定します。このプロパティーの値はオプションです。

デフォルト値はありません。

jms.TransportOptimized

jms.TransportOptimized プロパティーは、WIP (処理中の作業) 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。

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

デフォルト値は false です。

jms.UserName

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

デフォルト値はありません。

JvmMaxHeapSize

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

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

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

このプロパティーは、RepositoryDirectory 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 プロパティーは、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。このプロパティーにより、データベースへの複数イベントの書き込み操作をバッチ処理できるので、システム・パフォーマンスが向上します。

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

デフォルト値は 1 です。

Locale

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

ロケール名は、次の書式で指定します。

ll_TT.codeset

各変数の意味は、以下のとおりです。
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。詳しくは、本書の付録『Connector Configurator』を参照してください。

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

デフォルト値は `en_US` です。

LogAtInterchangeEnd

LogAtInterchangeEnd プロパティーは、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ーからアプリケーションへの接続が失われると、指定されたメッセージ宛先に、E メール・メッセージが送信されます。

このプロパティーは、RepositoryDirectory プロパティーの値が <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 プロパティーは、コントローラー・バッファー内のイベントの最大数を指定します。このプロパティーは、フロー制御機能によって使用されます。

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

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

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

デフォルト値は `InterchangeSystem` です。

MonitorQueue

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

このプロパティーは、DeliveryTransport プロパティーの値が `JMS` であり、DuplicateEventElimination の値が `true` である場合のみ有効です。

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

OADAutoRestartAgent

OADAutoRestartAgent プロパティーは、コネクターの使用する再始動機能が自動か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ーの再始動や System Monitor からのリモート・コネクターの始動を行います。

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、このプロパティーを `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 機能の構成方法の詳細については、「*システム・インストール・ガイド (Windows 版)*」または「*システム・インストール・ガイド (UNIX 版)*」を参照してください。

このプロパティーは、RepositoryDirectory プロパティーの値が <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 プロパティーは、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) がコネクターの再始動を自動的に試行する回数の最大数を指定します。OADAutoRestartAgent プロパティーを有効にするには、値を `true` に設定する必要があります。

このプロパティーは、RepositoryDirectory プロパティーの値が <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 プロパティーは、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。コネクター・エージェントがこの再試行間隔の間に再始動しないと、コネクター・コントローラーが OAD にコネクター・エージェントの再始動を再度要求します。OAD はこの再試行処理を OADMaxNumRetry プロパティーで指定されている回数だけ繰り返します。 OADAutoRestartAgent プロパティーを有効にするには、値を `true` に設定する必要があります。

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

このプロパティーには必ず有効な値を指定してください。デフォルト値は、値を含まない `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・ランタイムは、

- PollStartTime が設定され、PollEndTime が設定されていないこと
- PollEndTime が設定され、PollStartTime が設定されていないこと

を検出した場合、PollFrequency プロパティーに構成された値を使用してポーリングを実行します。

PollFrequency

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

- ポーリングし、PollQuantity プロパティーの値により指定される数のオブジェクトを取得します。

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

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

PollQuantity

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

このプロパティーは、`DeliveryTransport` プロパティーの値が `JMS` であり、`ContainerManagedEvents` プロパティーに値がある場合のみ有効です。

電子メール・メッセージもイベントと見なされます。コネクターは、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。

- 1 度ポーリングされると、コネクター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、コネクターはメッセージを無視します。
- コネクターは最初の 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、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
- 2 度目にポーリングされると、コネクターは 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、コネクターはビジネス・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 番目の添付ファイルが送信されます。

PollStartTime

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

このプロパティーには必ず有効な値を指定してください。デフォルト値は、値を含まない 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・ランタイムは、

- PollStartTime が設定され、PollEndTime が設定されていないこと
- PollEndTime が設定され、PollStartTime が設定されていないこと

を検出した場合、PollFrequency プロパティーに構成された値を使用してポーリングを実行します。

RepositoryDirectory

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

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

統合ブローカーが WebSphere Message Broker または WAS の場合は、この値はデフォルトで *<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 プロパティーは、統合ブローカーからコネクターへビジネス・オブジェクトが送信されるときに使用されるキューを指定します。

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

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

ResponseQueue

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

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

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

RestartRetryCount

RestartRetryCount プロパティーは、コネクターによるコネクター自体の再始動の試行回数を指定します。このプロパティーを並列に接続されたコネクターに対して使用する場合、コネクターのマスター側のアプリケーション固有のコンポーネントがクライアント側のアプリケーション固有のコンポーネント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 7 です。

RestartRetryInterval

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

プロパティー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。

ResultsSetEnabled

ResultsSetEnabled プロパティーは、Information Integrator がアクティブである場合、結果セット・サポートを使用可能または使用不可に設定します。このプロパティーは、アダプターが DB2 Information Integrator をサポートする場合にのみ有効です。

このプロパティーは、DeliveryTransport の値が JMS であり、 BrokerType の値が WMQI である場合のみ有効です。

デフォルト値は false です。

ResultsSetSize

ResultsSetSize プロパティーは、Information Integrator に戻すことが可能なビジネス・オブジェクトの最大数を定義します。このプロパティーは、アダプターが DB2 Information Integrator をサポートする場合にのみ有効です。

このプロパティーは、ResultsSetEnabled プロパティーの値が true の場合のみ有効です。

デフォルト値は 0 です。これは、結果セットのサイズが無制限であることを意味します。

RHF2MessageDomain

RHF2MessageDomain プロパティーにより、JMS ヘッダーのドメイン名フィールドの値を構成できます。JMS トランスポートを介してデータを WebSphere メッセージ・ブローカーに送信するときに、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>
```

このプロパティーは、BrokerType の値が WMQI または WAS である場合のみ有効です。また、このプロパティーは、DeliveryTransport プロパティーの値が JMS であり、WireFormat の値が CwXML である場合のみ有効です。

可能な値は、mrm および xml です。デフォルト値は mrm です。

SourceQueue

SourceQueue プロパティーは、JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、72 ページの『ContainerManagedEvents』を参照してください。

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

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

このプロパティーは、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 プロパティーは、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ー・フレームワークにデリバリーします。このキューは、コネクターが同期実行を使用する場合にのみ必要です。

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

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

TivoliMonitorTransactionPerformance

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

デフォルト値は false です。

WireFormat

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

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

WsifSynchronousRequestTimeout

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

このプロパティーは、 BrokerType の値が WAS の場合のみ有効です。

デフォルト値は 0 です。

XMLNameSpaceFormat

XMLNameSpaceFormat プロパティーは、ビジネス・オブジェクト定義の XML 形式内の短いネーム・スペースまたは長いネーム・スペースを指定します。

このプロパティーは、BrokerType の値が WMQI または WAS に設定されている場合のみ有効です。

デフォルト値は short です。

付録 B. Connector Configurator

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

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

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

この付録では、次のトピックについて説明します。

Connector Configurator の概要

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

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

ご使用のアダプターが DB2 Information Integrator をサポートする場合、WMQI オプションおよび DB2 II 標準プロパティーを使用します (付録『標準構成プロパティー』の注の欄を参照してください)。

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

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

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

Connector Configurator の実行モードと使用する構成ファイルのタイプは、実行する統合ブローカーによって異なる場合があります。例えば、使用している統合ブローカーが WMQI の場合、Connector Configurator を System Manager から実行するのではなく、直接実行します (89 ページの『スタンドアロン・モードでの Configurator の実行』を参照)。

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

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

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

ただし**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクター固有プロパティーのテンプレートを作成します。システム内で既にテンプレートが作成されている場合には、作成されているテンプレートを使用します。システム内でまだテンプレートが作成されていない場合には、90 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。

UNIX でのコネクターの実行

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

Connector Configurator のプロパティーには、ディレクトリー・パスを使用するものがあります。それらは、デフォルトでは Windows のディレクトリー・パスの規則を使用します。UNIX 環境で構成ファイルを使用する場合、UNIX のこれらのパスの規則に一致するようにディレクトリー・パスを修正してください。拡張検証に正しいオペレーティング・システム規則が使用されるように、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 の始動

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

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

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 を実行し、コネクター構成ファイルを編集できます。

これを行うには、以下のステップを実行します。

- 「**スタート**」>「**プログラム**」から、「**IBM WebSphere Business Integration Adapters**」>「**IBM WebSphere Business Integration Toolset**」>「**Connector Configurator**」をクリックします。
- 「**ファイル**」>「**新規**」>「**コネクター構成**」を選択します。
- 「**システム接続: 統合ブローカー**」の隣のプルダウン・メニューをクリックします。使用しているブローカーに応じて、ICS、WebSphere Message Brokers、WAS のいずれかを選択します。

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存してください (96 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

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

コネクター固有のプロパティー・テンプレートの作成

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

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

- テンプレートの新規作成については、90 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でこのテンプレートを保管します。既存のテンプレートは ¥WebSphereAdapters¥bin¥Data¥App ディレクトリーにあります。

新規テンプレートの作成

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

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

1. 「**ファイル**」>「**新規**」>「**コネクター固有プロパティー・テンプレート (Connector-Specific Property Template)**」とクリックします。
2. 「**コネクター固有プロパティー・テンプレート**」 ダイアログ・ボックスが表示されます。
   - 「**新規テンプレート名を入力してください**」の下の「**名前**」フィールドに、新規テンプレートの名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
   - テンプレートに含まれているコネクター固有のプロパティー定義を調べるには、「**テンプレート名**」表示でそのテンプレートの名前を選択します。そのテンプレートに含まれているプロパティー定義のリストが「**テンプレートのプレビュー**」表示に表示されます。
3. テンプレートを作成するときには、コネクターに必要なプロパティー定義に類似したプロパティー定義が含まれている既存のテンプレートを使用できます。ご使用のコネクターで使用するコネクター固有のプロパティーが表示されるテンプレートが見つからない場合は、自分で作成する必要があります。
   - 既存のテンプレートを変更する場合には、「**変更する既存のテンプレートを選択してください: 検索テンプレート**」の下の「**テンプレート名**」テーブルのリストから、テンプレート名を選択します。
   - このテーブルには、現在使用可能なすべてのテンプレートの名前が表示されます。テンプレートを検索することもできます。

一般特性の指定

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

- **一般:**
  プロパティー・タイプ

プロパティー・サブタイプ (Property Subtype)
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

「**プロパティー・タイプ**」 が String である場合に、「**プロパティー・サブタイプ (Property Subtype)**」を選択できます。構成ファイルの保管時に構文検査機能を提供するオプションの値です。デフォルトはブランク・スペースで、プロパティーのサブタイプが指定されていないことを意味します。

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

値の指定

「**値**」タブを使用すると、プロパティーの最大長、最大複数値、デフォルト値、または値の範囲を設定できます。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. 「**プロパティーを編集**」表示でプロパティーの名前を選択します。
3. 「**最大長**」および「**最大複数値**」のフィールドに値を入力します。

新規プロパティー値を作成するには、以下のステップを実行します。

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

「**値**」パネルには、3 つの列からなるテーブルが表示されます。

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

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

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

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

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

依存関係の設定

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

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

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

   == (等しい)

   != (等しくない)

   > (より大)

   < (より小)

   >= (より大か等しい)

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

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 UNIX 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 という形式に従う必要があります。例えば、C:¥WebSphereAdapters¥bin¥Data¥Std¥StdConnProps.xml とします。
  UNIX では、最初の文字は / でなければなりません。
- キュー名の先頭または途中に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。

ファイルでの拡張検証用に、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、プロパティーの拡張検証用のターゲット・オペレーティング・システムを選択できます。選択可能なオプションは、「Windows」、「UNIX」、および「その他」 (Windows でも UNIX 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 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**」を選択します。Connector Configurator が開きます。
- スタンドアロン・モードで、Connector Configurator 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**」>「**コネクター構成**」を選択します。「新規コネクター」ウィンドウで、新規コネクターの名前を入力します。

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

- 「**Integration Broker**」フィールドで、ICS 接続、WebSphere Message Brokers 接続、WAS 接続のいずれかを選択します。
- この章で後述する説明に従って「**新規コネクター**」ウィンドウの残りのフィールドを完了します。

コネクター固有のテンプレートからの構成ファイルの作成

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

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

コネクターの名前を入力します。名前では大文字と小文字が区別されます。入力する名前は、システムにインストールされているコネクターのファイル名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 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」、「WebSphere Message Brokers」、「WAS」のいずれかをクリックします。

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

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

  使用するテンプレートを選択し、**「OK」**をクリックします。

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

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

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

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

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

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

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

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

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

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

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

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

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

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

構成ファイルの完成

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

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

1. 「**標準のプロパティー**」タブで、BrokerType プロパティーの値フィールドを選択します。ドロップダウン・メニューで、値 ICS、WMQI、または WAS を選択します。
2. 選択したブローカーに関連付けられているコネクター・プロパティーが「標準のプロパティー」タブに表示されます。表に、「**プロパティー名**」、「**値**」、「**タイプ**」、「**サブタイプ**」 (「タイプ」が String 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 100 ページの『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。
4. 構成が完了したら、「**ファイル**」>「**保管**」>「**プロジェクトに**」を選択するか、または「**ファイル**」>「**保管**」>「**ファイルに**」を選択します。

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

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

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

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

   「**ターゲット・システム**」ドロップ・リストから「Windows」、「UNIX」、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き、または既存のコネクター構成ファイルを開くときには、Connector Configurator によって構成画面が表示されます。構成画面には、必要な構成値のカテゴリーに対応する複数のタブがあります。

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

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

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

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

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

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

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

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

「**標準のプロパティー**」と「**コネクター固有プロパティー (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かを示すために色分けされています。

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

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下の手順を実行します。

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

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

特定の標準プロパティーに関する詳細を参照するには、「標準のプロパティー」タブ付きシート内のそのプロパティーの「説明」列内の項目を左クリックします。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、標準プロパティーの詳細が表示されます。

注: ホット・ボタンが表示されない場合、そのプロパティーの全般ヘルプはありません。

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。

コネクター固有の構成プロパティーの設定

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

1. グリッドの左上端の部分で右クリックします。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ーを追加します。子プロパティーを追加するには、親行番号を右クリックして、「**子を追加**」をクリックします。
2. プロパティーまたは子プロパティーの値を入力します。

   注: プロパティーの「タイプ」が String 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、プロパティーの拡張検証に使用されます。
3. プロパティーを暗号化するには、「**暗号化**」ボックスを選択します。
4. 特定のプロパティーに関する詳細を参照するには、そのプロパティーの「説明」列内の項目を左クリックします。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、標準プロパティーの詳細が表示されます。

   注: ホット・ボタンが表示されない場合、そのプロパティーの全般ヘルプはありません。
5. 98 ページの『標準コネクター・プロパティーの設定』で説明したように、変更内容を保管するかまたは破棄するかを選択します。

全般ヘルプがインストール済みで、AdapterHelpName プロパティーがブランクである場合、Connector Configurator 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 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。付録『標準構成プロパティー』で説明されている AdapterHelpName プロパティーを参照してください。

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

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

コネクター・プロパティーの暗号化

「コネクター固有プロパティー」ウィンドウの「**暗号化**」チェック・ボックスにチェックマークを付けると、アプリケーション固有のプロパティーを暗号化することができます。値の暗号化を解除するには、「**暗号化**」チェック・ボックスをクリッ

クしてチェックマークを外し、「**検証**」 ダイアログ・ボックスに正しい値を入力し、「**OK**」 をクリックします。入力された値が正しい場合は、暗号化解除された値が表示されます。

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

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

更新メソッド

付録 『標準構成プロパティー』の 60 ページの『構成プロパティー値の概要』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

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

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

ご使用のブローカーが ICS の場合

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

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

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

たサポートを含む、変更されたコネクター定義が、System Manager の ICL (Integration Component Library) プロジェクトに保管されます。

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

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

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

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

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

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

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

ほとんどのコネクターの場合、選択可能な項目は「最大限の努力」のみです。

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

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

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

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

「**メッセージ・セット ID**」 は WebSphere Business Integration Message Broker 5.0 のオプション・フィールドで、指定されている場合一意である必要はありません。ただし、WebSphere MQ Integrator および Integrator Broker 2.1 では、一意の **ID** を指定する必要があります。

ご使用のブローカーが WAS の場合

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

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

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

関連付けられたマップ (ICS)

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

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

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

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

「**関連付けられたマップ**」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

  これらは、「**サポートされているビジネス・オブジェクト**」タブで指定した、このコネクターでサポートされるビジネス・オブジェクトです。「サポートされているビジネス・オブジェクト」タブで、サポートされるビジネス・オブジェクト

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

- **関連付けられたマップ**

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

- **明示的バインディング**

  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。ICS は、ブート時、コネクターごとに、サポートされる各ビジネス・オブジェクトにマップを自動的にバインドしようとします。複数のマップでその入力データとして同一のビジネス・オブジェクトが使用されている場合、サーバーは、他のマップのスーパーセットである 1 つのマップを見付けて、バインドしようとします。

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

  以下の手順を実行して、マップを明示的にバインドします。

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

リソース (ICS)

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

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

メッセージング (ICS)

「**メッセージング (Messaging)**」タブを使用すると、メッセージング・プロパティーを構成することができます。メッセージング・プロパティーは、`DeliveryTransport`

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

メッセージング・キューの検証

メッセージング・キューを検証するには、以下のようにする必要があります。

- WebSphere MQ Series がインストールされていることを確認する。
- ホスト・マシン上にチャネルおよびポートを持つメッセージング・キューを作成する。
- ホスト・マシンへの接続をセットアップする。

キューを検証するには、「メッセージング (Messaging)」タブの「メッセージング・タイプ (Messaging Type)」および「ホスト名 (Host Name)」フィールドの右にある「検証 (Validate)」ボタンを使用します。

セキュリティー (ICS)

Connector Configurator 内の「**セキュリティー**」タブを使用して、メッセージに対し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 プロパティーが 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`
- UNIX の場合:
  `opt/IBM/WebSphereAdapters/connectors/security/<connectorname>.jks`

このパスおよびファイルは、コネクター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 に設定されていない限り、これはグレー表示 (使用不可) となります。

「**メッセージのプライバシー・レベル**」は、3 つのメッセージ・カテゴリー (全メッセージ、全管理メッセージ、および全ビジネス・オブジェクト・メッセージ) 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・カテゴリーに対してプライバシー・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・カテゴリーに対して意図的にプライバシー・レベルなしと設定する場合に使用します。
- 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。
- 「セキュリティー」タブの「**鍵ストア・ターゲット・システムの絶対パス名**」で入力した値が、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て「Connector Configurator」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)。
- インポートの認証局がサーバー名になります。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

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

ログとトレースの値を変更するには、以下の手順を実行します。

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

     注: STDOUT オプションは、Windows プラットフォームで実行しているコネクターの「**トレース/ログ・ファイル**」タブでのみ使用できます。
   - ファイルに:
     ログ・メッセージまたはトレース・メッセージを指定されたファイルに書き込

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

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

データ・ハンドラー

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

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

構成ファイルの保管

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

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

- System Manager から、`*.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。
- WebSphere Application Server プロジェクトをセットアップしている場合には、このファイルを WebSphere Application Server プロジェクトに保管することもできます。

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

- ICS: 「*WebSphere InterChange Server システム・インプリメンテーション・ガイド*」

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

構成ファイルの変更

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

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

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

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

構成の完了

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

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

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

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

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

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

「CharacterEncoding」および「**ロケール**」標準構成プロパティーのドロップ・リストに表示されるのは、サポートされる値の一部のみです。ドロップ・リストに、サ

ポートされる他の値を追加するには、製品ディレクトリーの ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。

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

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

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

付録 C. チュートリアル

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 ユーティリティーの詳細については、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、(IBM WebSphere Business Integration 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・コネクター・リポジトリーを使用します。ポート・コネクター・リポジトリーは 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・コネクターはコネクター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 が入力キューに書き込んだ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 (同様に、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・コネクターをシミュレートし、Adapter for WebSphere MQ が発行したビジネス・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 Integrator 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 製品をインストール済みであり、その運用経験をもっていること。
- WebSphere MQ 5.1 以降をインストール済みであること。
- WebSphere MQ client libraries for Java をインストール済みであること。
- Adapter for WebSphere MQ をインストール済みであること (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・リポジトリーは *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 を使用して、*sample_folder*¥WebSphereMQConnector.cfg を開きます。Connector Configurator の使用法に関する詳細情報は、87 ページの『付録 B. Connector Configurator』を参照してください。コネクター固有プロパティーの詳細については、25 ページの『コネクター固有のプロパティー』を参照してください。
アダプター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ーを確認または変更します。
   - BrokerType このプロパティーを WMQI に設定します。
   - RepositoryDirectory このプロパティーを *sample_folder* ディレクトリーに設定します。

   以下のコネクター固有プロパティーを設定します。
   - ConfigurationMetaObject このプロパティーを Sample_WebSphereMQ_MO_Config に設定します。
   - DataHandlerConfigMO このプロパティーを Sample_WebSphereMQ_MO_DataHandler に設定します。
   - DataHandlerMimeType このプロパティーを text/delimited に設定します。
   - DataHandlerClassName このプロパティーを com.crossworlds.DataHandlers.text.delimited に設定します。
   - ErrorQueue このプロパティーを queue://crossworlds.queue.manager/MQCONN.ERROR に設定します。
   - InProgressQueue このプロパティーを queue://crossworlds.queue.manager/MQCONN.IN_PROGRESS に設定します。
   - InputQueue このプロパティーを queue://crossworlds.queue.manager/MQCONN.IN に設定します。
   - hostname このプロパティーをご使用のマシン名に設定します。
   - port このプロパティーを 1414 に設定します。
   - channel このプロパティーを CHANNEL1 に設定します。
   - UnsubscribedQueue このプロパティーを queue://crossworlds.queue.manager/MQCONN.UNSUBSCRIBED に設定します。

3. **ポート・コネクターの構成** Connector Configurator を使用して、以下の標準プロパティーを設定します。
   - BrokerType このプロパティーを WMQI に設定します。
   - RepositoryDirectory このプロパティーを *sample_folder* ディレクトリーに設定します。
   - RequestQueue このプロパティーを WebSphereMQConnector/DELIVERYQUEUE に設定します (WebSphere MQ アダプターの DeliveryQueue プロパティー値)。
   - DeliveryQueue このプロパティーを WebSphereMQConnector/REQUESTQUEUE に設定します (WebSphere MQ アダプターの RequestQueue プロパティー値)。
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 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 20 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 20. 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 を使用して、*sample_folder* 内にあるポート・コネクター定義 PortConnector.cfg を開きます。次に、表 21 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 21. ポート・コネク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_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. **コネクター開始スクリプトの作成または更新**

   Windows の場合

   a. Adapter for WebSphere MQ のショートカットのプロパティーを開きます。
   b. 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。例えば、以下のようになります。

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

   UNIX の場合

   a. ファイル *Product_Dir*/bin/connector_manager_WebSphereMQ を開きます。

b. AGENTCONFIG_FILE プロパティーを *<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。例えば、以下のようになります。

```
AGENTCONFIG_FILE=Product_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

チュートリアルの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、40 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、「コネクター・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・コネクター構成ファイル `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ーが必要です。このようなユーティリティーが使用できない場合は、WebSphere アダプターの `InputQueue` プロパティーを `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、43 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、110 ページの『始める前に』を参照してください。さらに、113 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 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、「コネクター・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・コネクター構成ファイル 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。

付録 D. Common Event Infrastructure

WebSphere Business Integration Server Foundation には、Common Event Infrastructure が作動するために必要な Common Event Infrastructure Server Application が含まれています。WebSphere Application Server Foundation は、任意のシステム (アダプターがインストールされるマシンと同一のマシンである必要はありません) にインストールできます。

WebSphere Application Server Application Client には、アダプターと Common Event Infrastructure Server Application の間の対話のために必要なライブラリーが含まれています。アダプターをインストールしたシステムと同一のシステムに WebSphere Application Server Application Client をインストールする必要があります。アダプターは、構成可能な URL を用いて (WebSphere Business Integration Server Foundation 内において) WebSphere Application Server に接続します。

今回のリリースでサポートされるどの統合ブローカーを使用しても、Common Event Infrastructure サポートが使用可能です。

必要なソフトウェア

Common Event Infrastructure が作動するためには、アダプターに必要なソフトウェア前提条件に加え、以下のソフトウェアがインストールされていなければなりません。

- WebSphere Business Integration Server Foundation 5.1.1
- WebSphere Application Server Application Client 5.0.2、5.1、または 5.1.1

  (WebSphere Application Server Application Client 5.1.1 は、WebSphere Business Integration Server Foundation 5.1.1 に付属しています。)

注: Common Event Infrastructure は、HP-UX または Linux プラットフォームではサポートされていません。

Common Event Infrastructure の使用可能化

Common Event Infrastructure 機能は、Connector Configurator を用いて構成される標準プロパティー `CommonEventInfrastructure` および `CommonEventInfrastructureContextURL` によって使用可能になります。デフォルトでは、Common Event Infrastructure は使用不可です。`CommonEventInfrastructureContextURL` プロパティーによって、Common Event Infrastructure サーバーの URL を構成することができます。(詳細については、本書の付録『標準構成プロパティー』を参照してください。)

Common Event Infrastructure アダプター・イベントの取得

Common Event Infrastructure が使用可能である場合、アダプターは、以下のアダプター・イベントにマップする Common Event Infrastructure イベントを生成します。

- アダプターの開始
- アダプターの停止
- アダプター・エージェントからタイムアウトへのアプリケーション応答
- アダプター・エージェントから発行された `doVerbFor` 呼び出し
- アダプター・エージェントからの `gotApplEvent` 呼び出し

別のアプリケーション (「コンシューマー・アプリケーション」) が、アダプターの生成した Common Event Infrastructure イベントを受け取るためには、そのアプリケーションが Common Event Infrastructure カタログを使用して適切なイベントおよびそのプロパティーの定義を判別する必要があります。コンシューマー・アプリケーションが送信側アプリケーションのイベントを消費できるようにするには、イベント・カタログでイベントが定義されていなければなりません。

本書の付録『Common Event Infrastructure イベント・カタログ定義』には、WebSphere Business Information アダプターの場合、コンシューマー・アプリケーションが検索するイベント記述子およびプロパティーを示す XML 形式のメタデータが含まれています。

詳細情報

Common Event Infrastructure の詳細については、以下の URL で提供されている WebSphere Business Integration Server Foundation 資料内の Common Event Infrastructure 情報を参照してください。

http://publib.boulder.ibm.com/infocenter/ws51help

コンシューマー・アプリケーションの検索対象となる、アダプターによって生成されるイベント記述子およびプロパティーを示すサンプル XML メタデータについては、『Common Event Infrastructure イベント・カタログ定義』を参照してください。

Common Event Infrastructure イベント・カタログ定義

Common Event Infrastructure イベント・カタログには、他のアプリケーションによる照会が可能なイベント定義が含まれています。 XML メタデータを使用した、典型的なアダプター・イベントのイベント定義のサンプルを以下に示します。別のアプリケーションを作成する場合、そのアプリケーションで、イベント・カタログ・インターフェースを使用して、イベント定義に対する照会を行うことができます。イベント定義およびそれらの照会方法について詳しくは、オンラインの IBM WebSphere Server Foundation Information Center で提供されている Common Event Infrastructure 資料を参照してください。

WebSphere Business Integration アダプターでは、イベント・カタログで定義される必要のある拡張データ・エレメントが、ビジネス・オブジェクトのキーです。各ビジネス・オブジェクト・キーには、イベント定義が必要です。そのためどのアダプ

ターでも、start adapter、stop adapter、timeout adapter、および任意の doVerbFor イベント (例えば create、update、または delete) などのさまざまなイベントについて、対応するイベント定義がイベント・カタログ内に存在する必要があります。

以下のセクションで、start adapter、stop adapter、およびイベントの request または delivery の XML メタデータの例を示します。

「start adapter」メタデータの XML 形式

```
<eventDefinition name="startADAPTER"
          parent="event">
    <property name ="creationTime" //Comment: example value would be
"2004-05-13T17:00:16.319Z"
         required="true" />
    <property name="globalInstanceId" //Comment: Automatically generated
 by Common Event Infrastructure
          required="true"/>
    <property name="sequenceNumber"    //Comment: Source defined number
for messages to be sent/sorted logically
         required="false"/>
    <property name="version"    //Comment: Version of the event
          required="false"
         defaultValue="1.0.1"/>
    <property name="sourceComponentId"
         path="sourceComponentId"
          required="true"/>
    <property name="application"    //Comment: The name#version of the
source application generating the event. Example is "SampleConnector#3.0.0"
         path="sourceComponentId/application"         required="false"/>
    <property name="component"   //Comment: This will be the name#version
of the source component.
         path="sourceComponentId/component"
           required="true"
         defaultValue="ConnectorFrameWorkVersion#4.2.2"/>
    <property name="componentIdType"     //Comment: specifies the format
 and meaning of the component
         path="sourceComponentId/componentIdType"
           required="true"
         defaultValue="Application"/>
    <property name="executionEnvironment"
 //Comment: Identifies the environment the application is running
 in...example is "Windows 2000#5.0"
         path="sourceComponentId/executionEnvironment"
         required="false" />
    <property name="location"   //Comment: The value of this is the
server name...example is "WQMI"
          path="sourceComponentId/location"
          required="true"/>
     <property name="locationType" //Comment specifies the format and
 meaning of the location
          path="sourceComponentId/locationType"
           required="true"
          defaultValue="Hostname"/>
     <property name="subComponent"     //Comment:further distinction
of the logical component
          path="sourceComponentId/subComponent"
           required="true"
          defaultValue="AppSide_Connector.AgentBusinessObjectManager"/>
     <property name="componentType"      //Comment: well-defined name
used to characterize all instances of this component
          path="sourceComponentId/componentType"
           required="true"
          defaultValue="ADAPTER"/>
     <property name="situation" //Comment: Defines the type of
```

```
situation that caused the event to be reported
          path="situation"
          required="true"/>
     <property name="categoryName="    //Comment: Specifies the type
 of situation for the event
              path="situation/categoryName"
           required="true"
         defaultValue="StartSituation"/>
    <property name="situationType"    //Comment: Specifies the type
of situation and disposition of the event
         path="situation/situationType"
           required="true"
    <property name="reasoningScope" //Comment: Specifies the scope
of the impact of the event
         path="situation/situationType/reasoningScope"
           required="true"
         permittedValue="INTERNAL"
         permittedValue="EXTERNAL"/>
    <property name="successDisposition" //Comment: Specifies the
 success of event
         path="situation/situationType/successDisposition"
           required="true"
         permittedValue="SUCCESSFUL"
         permittedValue="UNSUCCESSFUL" />
    <property name="situationQualifier"  //Comment: Specifies the
 situation qualifiers for this event
           path="situation/situationType/situationQualifier"
           required="true"
         permittedValue="START_INITIATED"
         permittedValue="RESTART_INITIATED"
         permittedValue="START_COMPLETED" />
</eventDefinition>
```

「stop adapter」メタデータの XML 形式

「stop adapter」のメタデータは、以下の例外を除き「start adapter」のメタデータと同じです。

- categoryName プロパティーのデフォルト値は、StopSituation です。

```
<property name="categoryName="
 //Comment: Specifies the type
 of situation for the event

                            path="situation/categoryName"
          required="true"
             defaultValue="StopSituation"/>
```

- situationQualifier プロパティーで許可される値は異なり、「stop adapter」では以下のようになります。

```
<property name="situationQualifier"
 //Comment: Specifies the situation qualifiers for this event

                       path="situation/situationType/situationQualifier"
          required="true"
          permittedValue="STOP_INITIATED"
          permittedValue="ABORT_INITIATED"
          permittedValue="PAUSE_INITIATED"
          permittedValue="STOP_COMPLETED"
 />
```

「timeout adapter」メタデータの XML 形式

「timeout adapter」のメタデータは、以下の例外を除き「start adapter」および「stop adapter」のメタデータと同じです。

- categoryName プロパティーのデフォルト値は、ConnectSituation です。

```
<property name="categoryName="
 //Comment: Specifies the type
 of situation for the event
             path="situation/categoryName"
          required="true"
             defaultValue="ConnectSituation"/>
```

- situationQualifier プロパティーで許可される値は異なり、「timeout adapter」では以下のようになります。

```
<property name="situationQualifier"  //Comment: Specifies
 the situation qualifiers for this event

          path="situation/situationType/situationQualifier"
          required="true"
          permittedValue="IN_USE"
          permittedValue="FREED"
          permittedValue="CLOSED"
          permittedValue="AVAILABLE"
 />
```

「request」または「delivery」メタデータの XML 形式

この XML 形式の末尾には、拡張データ・エレメントがあります。 adapter request および delivery イベントの拡張データ・エレメントは、処理されているビジネス・オブジェクトからのデータを表します。このデータには、ビジネス・オブジェクトの名前、ビジネス・オブジェクトのキー (外部またはローカル)、および親ビジネス・オブジェクトの子であるビジネス・オブジェクトが含まれます。子ビジネス・オブジェクトは、次に親と同じデータに分解されます (名前、キー、および子ビジネス・オブジェクト)。このデータは、イベント定義の拡張データ・エレメントで表されます。このデータは、処理されているビジネス・オブジェクト、キー、子ビジネス・オブジェクトに応じて変化します。このイベント定義内の拡張データは、1 つの例に過ぎず、キー EmployeeId と、キー EmployeeId を持つ子ビジネス・オブジェクト EmployeeAddress を含む、Employee という名前のビジネス・オブジェクトを表します。このパターンは、特定のビジネス・オブジェクトに存在するデータと同じ量だけ継続する可能性があります。

```
<eventDefinition name="createEmployee"     //Comment: This
 extension name is always the business object verb followed by the business
 object name
          parent="event">
    <property name ="creationTime"  //Comment: example value would be
"2004-05-13T17:00:16.319Z"
         required="true" />
    <property name="globalInstanceId" //Comment: Automatically generated
 by Common Event Infrastructure
          required="true"/>
    <property name="localInstanceId"    //Comment: Value is business
 object verb+business object name+#+app name+ business object identifier
         required="false"/>
    <property name="sequenceNumber"    //Comment: Source defined number
for messages to be sent/sorted logically
         required="false"/>
    <property name="version"  //Comment: Version of the event...value is
 set to 1.0.1
```

```
          required="false"
         defaultValue="1.0.1"/>
    <property name="sourceComponentId"
         path="sourceComponentId"
          required="true"/>
    <property name="application"    //Comment: The name#version of the
 source application generating the event...example is
 "SampleConnector#3.0.0"
         path="sourceComponentId/application"
         required="false"/>
    <property name="component"   //Comment: This will be the name#version
of the source component.
         path="sourceComponentId/component"
           required="true"
         defaultValue="ConnectorFrameWorkVersion#4.2.2"/>
    <property name="componentIdType"     //Comment: specifies the format
 and meaning of the component
         path="sourceComponentId/componentIdType"
           required="true"
         defaultValue="Application"/>
    <property name="executionEnvironment" //Comment: Identifies the
 environment#version the app is running in...example is "Windows 2000#5.0"
         path="sourceComponentId/executionEnvironment"
         required="false" />
    <property name="instanceId" //Comment: Value is business object
  verb+business object name+#+app name+ business object identifier
          path="sourceComponentId/instanceId"
          required="false"
    <property name="location"   //Comment: The value of this is the
server name...example is "WQMI"
          path="sourceComponentId/location"
          required="true"/>
     <property name="locationType" //Comment specifies the format and
 meaning of the location
          path="sourceComponentId/locationType"
           required="true"
          defaultValue="Hostname"/>
     <property name="subComponent"  //Comment:further distinction of the
 logical component-in this case the value is the name of the business
 object
          path="sourceComponentId/subComponent"
          required="true"/>
     <property name="componentType"      //Comment: well-defined name used
 to characterize all instances of this component
          path="sourceComponentId/componentType"
           required="true"
          defaultValue="ADAPTER"/>
     <property name="situation" //Comment: Defines the type of
situation that caused the event to be reported
          path="situation"
          required="true"/>
    <property name="categoryName"    //Comment: Specifies the type
 of situation for the event
              path="situation/categoryName"
            required="true"
         permittedValue="CreateSituation"
         permittedValue="DestroySituation"
         permittedValue="OtherSituation" />
    <property name="situationType"    //Comment: Specifies the type
of situation and disposition of the event
         path="situation/situationType"
           required="true"
    <property name="reasoningScope" //Comment: Specifies the scope
of the impact of the event
         path="situation/situationType/reasoningScope"
           required="true"
         permittedValue="INTERNAL"
```

```
        permittedValue="EXTERNAL"/>
   <property name="successDisposition" //Comment: Specifies the
 success of event
        path="situation/situationType/successDisposition"
          required="true"
        permittedValue="SUCCESSFUL"
        permittedValue="UNSUCCESSFUL" />
   <extendedDataElements name="Employee" //Comment: name of business
 object itself
            type="noValue"
                   <children name="EmployeeId"
                 type="string"/>  //Comment: type is one of the
 permitted values within Common Event Infrastructure documentation
            <children name="EmployeeAddress"
                 type="noValue"/>
                   <children name="EmployeeId"
                       type="string"/>
                    -
                    -
                    -
   </extendedDataElements
</eventDefinition>
```

付録 E. Application Response Measurement

このアダプターは、Application Response Measurement アプリケーション・プログラミング・インターフェース (API) との互換性があります。この API によって、アプリケーションの可用性、サービス・レベル・アグリーメント、およびキャパシティー・プランニングを管理することができます。ARM を備えたアプリケーションは、IBM Tivoli Monitoring for Transaction Performance に参加することができ、これにより、トランザクションのメトリックに関するデータの収集および検討が可能です。

Application Response Measurement 計測機能サポート

このアダプターは、Application Response Measurement アプリケーション・プログラミング・インターフェース (API) との互換性があります。この API によって、アプリケーションの可用性、サービス・レベル・アグリーメント、およびキャパシティー・プランニングを管理することができます。ARM を備えたアプリケーションは、IBM Tivoli Monitoring for Transaction Performance に参加することができ、これにより、トランザクションのメトリックに関するデータの収集および検討が可能です。

必要なソフトウェア

ARM が作動するためには、アダプターに必要なソフトウェア前提条件に加え、以下のソフトウェアがインストールされていなければなりません。

- WebSphere Application Server 5.0.1 (IBM Tivoli Monitoring for Transaction Performance サーバーを含む)。これは、アダプターと同一のシステム上にインストールする必要はありません。
- IBM Tivoli Monitoring for Transaction Performance v. 5.2 フィックスパック 1。これは、アダプターがインストールされているシステムと同一のシステムにインストールし、IBM Tivoli Monitoring for Transaction Performance サーバーが存在するシステムを指すように構成する必要があります。

今回のリリースでサポートされるどの統合ブローカーを使用しても、Application Response Measurement サポートが使用可能です。

注: Application Response Measurement 計測機能は、HP-UX (すべてのバージョン) および Red Hat Linux 3.0 を除き、このリリースの IBM WebSphere Business Integration Adapters でサポートされるすべてのオペレーティング・システムでサポートされています。

Application Response Measurement の使用可能化

ARM 計測機能は、Connector Configurator 内の標準プロパティー `TivoliMonitorTransactionPerformance` を「True」に設定することによって、使用可能になります。デフォルトでは、標準では、ARM サポートは使用不可です。(詳細については、本書の付録『標準構成プロパティー』を参照してください。)

トランザクションのモニター

ARM が使用可能である場合、モニターされるトランザクションは、サービス要求およびイベント・デリバリーです。トランザクションは、サービス要求またはイベント・デリバリーの開始から、サービス要求またはイベント・デリバリーの終了まで計測されます。Tivoli Monitoring for Transaction Performance コンソールに表示されるトランザクションの名前は、SERVICE REQUEST または EVENT DELIVERY のいずれかで始まります。名前の次の部分は、ビジネス・オブジェクト動詞 (CREATE、RETRIEVE、UPDATE または DELETE など) になります。名前の最後の部分は、「EMPLOYEE」などのビジネス・オブジェクト名になります。例えば、従業員作成のイベント・デリバリー用のトランザクションの名前は、EVENT DELIVERY CREATE EMPLOYEE のようになります。別の例としては、SERVICE REQUEST UPDATE ORDER のようになります。

以下のメトリックは、デフォルトで、サービス要求またはイベント・デリバリーの各タイプごとに収集されます。

- 最小トランザクション時間
- 最大トランザクション時間
- 平均トランザクション時間
- トランザクション実行の合計

ユーザー (または WebSphere Application Server のシステム管理者) は、Tivoli Monitoring for Transaction Performance コンソール内から、ディスカバリー・ポリシーおよびリスナー・ポリシーを構成することによって、特定のトランザクションのどのメトリックを表示するか、どのアダプター・イベント用であるかを選択できます。(『詳細情報』を参照してください。)

詳細情報

詳しくは、IBM Tivoli Monitoring for Transaction Performance の資料を参照してください。特に、アダプターが生成するメトリックのモニターおよび管理の詳細については、「*IBM Tivoli Monitoring for Transaction Performance User's Guide*」を参照してください。

付録 F. サンプル・シナリオ

以下のサンプル・シナリオは、その順序に従ってインストールおよび構成する必要があります。サンプル・シナリオ 1 は、InterChange Server をブローカーとして使用した、WebSphere MQ および Virtual Test Connector 間の単純なビジネス・オブジェクト交換を示します。サンプル・シナリオ 2 は、動的子メタオブジェクトを使用してビジネス・オブジェクトを転送する方法を示します。

サンプル・シナリオ 1: ビジネス・オブジェクトの送受信

サンプル・シナリオ 1 は、WBI Adapter for WebSphere MQ を使用して、WebSphere MQ を使用して通信を行うアプリケーションとの間でビジネス・オブジェクトの送受信を行う方法を示します。このシナリオは、単純になるように設計され、アダプターの基本的な機能を示します。

要約

この架空のシナリオには、作成、更新、または削除される企業の連絡に関する情報を交換することができるアプリケーション (″ApplicationX″) が関わっています。

ApplicationX からのメッセージに定義されたフィールドと一致するビジネス・オブジェクト Sample_WebSphereMQ_LegacyContact を作成します。幸いなことに、この架空のシナリオでは、ApplicationX が送受信するメッセージのフォーマットは、WBI EDK に付属する区切りデータ・ハンドラーに準拠しています。

WebSphere MQ コネクターおよびポート・コネクター間で「Sample_MQSeries_LegacyContact」ビジネス・オブジェクトを送達するために必要な、テンプレートおよびコラボレーションを作成します。ポート・コネクターは、IBM WebSphere Business Integration 製品のすべてのインストール済み環境に含まれています。これは、基礎となるコードがなくコネクター定義のみで構成されているため、使用するコラボレーションの「ダミー」コネクターとして機能します。このコネクターは、容易に JText などの別のコネクターに変更できます (そのように選択した場合)。

始動したアダプターは、ApplicationX が入力キューに書き込んだ連絡メッセージを検索します。アダプターは Delimited DataHandler 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、IBM WebSphere Interchange Server (ICS) ブローカーに送達します。Test Connector (同様に、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・コネクターをシミュレートし、アダプターが発行したビジネス・オブジェクトを検索し、属性を確認します。データを変更してから、メッセージをブローカーに再送達します。ここからメッセージはアダプターに送信され、メッセージに変換され、ApplicationX の入力キューに送達されます。

この架空のシナリオは、統合ブローカーとしての ICS 用に構成されたアダプターを使用して示されます。

前提事項

サンプル・シナリオをインストールおよび実行するには、以下の要件を満たす必要があります。

- WebSphere ICS 4.2.x 以降の製品をインストール済みであり、その運用経験をもっていること。
- IBM WebSphere MQ 5.1 以上をインストール済みであり、その運用経験をもっていること。
- WebSphere MQ client libraries for Java をインストール済みであること。
- WebSphere Business Integration Adapter for WebSphere MQ を WebSphere ICS ホーム・ディレクトリーにインストール済みであること。
- WebSphere MQ 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。

サンプル・シナリオのインストールおよび構成

このテスト・シナリオでは、InterchangeServer (ICS) ブローカー・タイプ構成の場合の、WebSphere MQ アダプターおよび VTC の間の単純なビジネス・オブジェクト交換を示します。以下のステップに従って、サンプル・シナリオをインストールおよび構成します。

注 1: 本書で %SAMPLE_FOLDER% と記述されている箇所は、本書が格納されているフォルダーを指しています。

注 2: 本書で %CROSSWORLDS% と記述されている箇所は、現在の WebSphere ICS インストール済み環境を格納するフォルダーを指しています。すべての環境変数およびファイル区切りは、Windows NT/2000 フォーマットで指定されます。UNIX で実行する場合は、適切な変更を加えてください (例えば、%CROSSWORLDS%¥connectors は ${CROSSWORLDS}/connectors となります)。

1. キューを作成します。
   このサンプル・シナリオでは、キュー・マネージャーに 3 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.ERROR')

   DEFINE QL('LEGACYAPP.IN')
   ```

   注 1: MO には、`OutputQueue=LGACYAPP.IN` の ASI を含むデフォルト属性があります (URI 形式)。これを、適切なキュー・マネージャー名に変更することができます。

   注 2: `LEGACYAPP.IN` は、アプリケーションの入力キューであり、アダプターの出力キューと同一です。MO のデフォルト属性で構成されます。

2. InterChange Server および System Manager を始動します。
   ショートカットから WebSphere Interchange Server を始動します。 WebSphere Business Integration System Manager を始動し、「コンポーネント・ナビゲーター・パースペクティブ (Component Navigator Perspective)」を開きます。
   「Interchange Servers」ビューで、ご使用のサーバーをサーバー・インスタンスとして登録および接続します。
3. WebSphere MQ サンプル・ソリューションをロードします。
   「コンポーネント・ナビゲーター・パースペクティブ (Component Navigator Perspective)」から、新規の統合コンポーネント・ライブラリーを作成し、%CROSSWORLDS%¥connectors¥WebSphereMQ¥samples¥WebSphereICS¥ にある WebSphereMQSample.jar という名前の repos ファイルをインポートします。
4. コラボレーション・テンプレートをコンパイルします。
   WebSphere Business Integration System Manager を使用して、「コラボレーション・テンプレート」というラベルが付けられたフォルダーを右クリックし、次にドロップダウン・リストから「すべてコンパイル」を選択します。
5. アダプターを構成します。
   WebSphere Business Integration System Manager を使用して、Connector Designer 内で WebSphere MQ アダプターを開きます。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。また、アダプター構成プロパティーが以下に示す値に一致することを確認してください。
   - **標準プロパティー**:
     - プロパティー 「BrokerType」の値を「ICS」に設定します。
     - プロパティー 「DeliveryTransport」の値を「IDL」に設定します。
   - **コネクター固有のプロパティー:**
     - プロパティー「ConfigurationMetaObject」を値「Sample_WebSphereMQ_MO_Config」に設定します。
     - プロパティー「DataHandlerConfigMO」を値「Sample_WebSphereMQ_MO_DataHandler」に設定します。
     - プロパティー「DataHandlerMimeType」を値「text/delimited」に設定します。
     - プロパティー「DataHandlerClassName」を値「com.crossworlds.DataHandlers.text.delimited」に設定します。
     - プロパティー「ErrorQueue」を値「queue://crossworlds.queue.manager/MQCONN.ERROR」に設定します。
     - プロパティー「InputQueue」を値「queue://crossworlds.queue.manager/MQCONN.IN」に設定します。
     - 「hostname」プロパティーをご使用のマシンの名前に設定します。
     - 「port」プロパティーを 1414 に設定します。
     - 「channel」プロパティーを CHANNEL1 に設定します。
     - プロパティー「UnsubscribedQueue」を値「queue://crossworlds.queue.manager/MQCONN.UNSUBSCRIBED」に設定します。

- **サポートされているビジネス・オブジェクト**:
  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 Designer を使用して、「サポートされているビジネス・オブジェクト」タブを選択し、「ビジネス・オブジェクト名」ドロップダウン・リストから以下のビジネス・オブジェクトを選択してから、エージェントがサポートするチェック・ボックスにチェック・マークを付けます。
  - ″Sample_WebSphereMQ_MO_Config″
  - ″Sample_WebSphereMQ_MO_DataHandler″
  - ″Sample_WebSphereMQ_LegacyContact″

6. ポート・コネクターを構成します。
   WebSphere Business Integration System Manager を使用して、Connector Designer 内で PortConnector を開きます。
   - **標準プロパティー**:
     - プロパティー「BrokerType」の値を「ICS」に設定します。
     - プロパティー 「DeliveryTransport」の値を「IDL」に設定します。
   - **サポートされているビジネス・オブジェクト**:
     Connector Designer を使用して、「サポートされているビジネス・オブジェクト」タブを選択し、ドロップダウン・リストから以下のビジネス・オブジェクトを選択してから、エージェントがサポートするビジネス・オブジェクトとしてチェック・マークを付けます。
     - ″Sample_WebSphereMQ_LegacyContact″

     %SAMPLE_FOLDER% に、ポート・コネクター構成ファイルを「PortConnector.cfg」として保管します。
7. メタオブジェクトを構成します。
   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。

   重要: 必ず Connector Configurator 内の変更内容を保管してください。
8. WebSphere MQ ユーザー・プロジェクトを作成します。
   WebSphere Business Integration System Manager を使用して、新規ユーザー・プロジェクトを作成します。ステップ 3 (129 ページ) で作成された統合コンポーネント・ライブラリーから、すべてのコンポーネントを選択します。
9. ユーザー・プロジェクトを InterChange Server に追加して配置します。
   「サーバー・インスタンス (Server Instance)」ビューから、ステップ 8 で作成したユーザー・プロジェクトを WebSphere ICS に追加し、次にこのユーザー・プロジェクトからすべてのコンポーネントを InterChange Server に配置します。
10. InterChange Server をリブートします。
    すべての変更を有効にするには、InterChange Server をリブートします。System Monitor ツールを使用して、すべてのコラボレーション・オブジェクト、コネクター・コントローラー、およびマップが緑色の状態になっていることを確認してください。

サンプル・シナリオの実行

以下のステップに従って、サンプル・シナリオを実行します。

1. アダプター がまだ稼働していない場合は始動します。
2. Visual Test Connector (VTC) がまだ稼働していない場合は始動します。
3. ポート・コネクターをシミュレートします。以下のステップに従って、これを実行します。
   a. VTC を使用して、「PortConnector」のプロファイルを定義します。以下のステップに従って、これを実行します。
      1) VTC メニューから、「ファイル」>「プロファイルを作成/選択」を選択します。
      2) 次に、「コネクター・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。
      3) %SAMPLE% ディレクトリーで ポート・コネクター構成ファイル「PortConnector.cfg」を選択します。
      4) 「Connector Name」および「Broker Type」を構成し、「OK」を選択します。
      5) 作成したプロファイルを選択してから、「OK」を選択します。
   b. 「ファイル」>「接続」を選択して、コネクター・エージェントのシミュレートを開始します。
4. イベントの消費をテストします。
   VTC 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。必要に応じてデフォルト値を変更し、メッセージを送信します。
5. メッセージが送達されたことを確認します。
   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。
6. サブスクリプション送達をテストします。

   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ーが必要です。このようなユーティリティーが使用できない場合は、アダプターの「InputQueue」プロパティーを「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、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。

実行の結果

上記のステップがすべて正常に実行された場合には、正常に機能するサンプル・シナリオが得られ、このシナリオにより WBI Adapter for WebSphere MQ 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。

サンプル・シナリオ 2: ビジネス・オブジェクトの転送

サンプル・シナリオ 2 は、DynamicMetaObject を使用して、WebSphere MQ で定義されたさまざまなキューへ BO を転送する方法を示します。

要約

このシナリオでは、Sample_WebSphereMQ_LegacyContact 内の子メタオブジェクトの属性を作成します。ビジネス・オブジェクトを WBI Adapter for WebSphere MQ に送信する場合、この子メタオブジェクト内の outputQueue の値を変更して、Sample_WebSphereMQ_LegacyContact ビジネス・オブジェクトを、選択したさまざまなキューへ転送します。

前提事項

サンプル・シナリオをインストールおよび実行するには、以下の要件を満たす必要があります。

- WebSphere ICS 4.2.x 以降の製品をインストール済みであり、その運用経験をもっていること。
- IBM WebSphere MQ 5.1 以上をインストール済みであり、その運用経験をもっていること。
- WebSphere MQ client libraries for Java をインストール済みであること。
- WBI Adapter for WebSphere MQ をインストール済みであること。
- WebSphere MQ 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。
- WebSphere MQ アダプターおよび PortConnector がセットアップ済みで、セクション 128 ページの『サンプル・シナリオのインストールおよび構成』で説明されているように、すべてのサンプル・コラボレーションおよびサンプル・オブジェクトがロードされていること。

動的メタオブジェクトのサンプル・シナリオのインストール

以下のステップに従って、動的メタオブジェクトのサンプル・シナリオをインストールします。

重要: サンプル・シナリオ 2 をインストールする前にサンプル・シナリオ 1 をインストールする必要があります。

1. アプリケーション固有の情報を追加して、動的メタオブジェクトを含む属性を識別します。
   Sample_WebSphereMQ_LegacyContact ビジネス・オブジェクトで、ビジネス・オブジェクト・レベルのアプリケーション固有情報に、「cw_mo_conn=DynMO」というエントリーを追加します。これにより属性が識別されます。
2. 実際の属性を親オブジェクトに追加します。以下のステップに従って、これを実行します。
   a. WebSphere Business Integration System Manager を使用して、以下のオブジェクトを開きます。
      - Sample_WebSphereMQ_DynMO_Config.xsd
      - Sample_WebSphereMQ_LegacyContact.xsd
   b. 「Sample_WebSphereMQ_LegacyContact Object」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. 新規ターゲット・キューを定義します。
   WebSphere MQ で一時キュー「REROUTE.IN」を定義します。これは、作成した Sample_WebSphereMQ_LegacyContact ビジネス・オブジェクトを転送する場所です。必要なキューを作成するには、コマンド行から「RUNMQSC crossworlds.queue.manager」と入力してから、以下のコマンドを発行します。
   `DEFINE QL('REROUTE.IN')`

動的メタオブジェクトのサンプル・シナリオの実行

以下のステップに従って、動的メタオブジェクトのサンプル・シナリオを実行します。

1. WBI 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。
3. ポート・コネクターをシミュレートします。以下のステップに従って、これを実行します。
   a. VTC を使用して、「PortConnector」のプロファイルを定義します。以下のステップに従って、これを実行します。
      1) VTC メニューから、「ファイル」>「プロファイルを作成/選択」を選択します。
      2) 次に、「コネクター・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。
      3) %SAMPLES% ディレクトリーで ポート・コネクター構成ファイル「PortConnector.cfg」を選択します。
      4) 「Connector Name」および「Broker Type」を構成し、「OK」を選択します。

5) 作成したプロファイルを選択してから、「OK」を選択します。

b. 「ファイル」>「接続」を選択して、コネクター・エージェントのシミュレートを開始します。

4. 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します。
   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。必要に応じてデフォルト値を変更します。DynMO 属性を右クリックして、そのインスタンスを作成することにより、Sample_WebSphereMQ_DynMO_Config のインスタンスも作成します。

5. 新規ターゲット・キュー (REROUT.IN) を設定します。
   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。outputQueue という名前の属性に、BO の行き先とするキューの名前を入力します。このサンプルの場合、新規の「REROUTE.IN」キューです。以下のように完全な URI を入力してください。
   `queue://<queue manager>/REROUTE.IN?targetClient=1`

6. ビジネス・オブジェクトを送信します。

7. メッセージが送達されたことを確認します。
   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、アダプターから新規連絡メッセージが到着したかどうかを確認します。

実行の結果

新しいメッセージがアダプターから 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できるようになりました。その場合、キュー名はビジネス・オブジェクト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で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

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

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

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

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

IBM 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が、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

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

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

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のめかしたり、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。

i5/OS
IBM
IBM ロゴ
AIX
AIX 5L
CICS
CrossWorlds
DB2
DB2 Universal Database
HelpNow
IMS
Informix
iSeries
Lotus
Lotus Domino
Lotus Notes
MQIntegrator
MQSeries
MVS
OS/400
Passport Advantage
pSeries
Redbooks
SupportPac
WebSphere
z/OS

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。

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。

Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Intel Centrino、Intel Centrino ロゴ、Celeron、Intel Xeon、Intel SpeedStep、Itanium、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。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

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。

この製品には、Eclipse Project (http://www.eclipse.org/) により開発されたソフトウェアが含まれています。

WebSphere Business Integration Adapter Framework バージョン 2.6.0.3

索引

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

[ア行]

[カ行]

[サ行]

[タ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[数字]

A

Printed in Japan

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