IBM WebSphere Business Integration
Adapters

Adapter for WebSphere MQ Integrator Broker ユーザーズ・ガイド

V 2.5.x

IBM WebSphere Business Integration
Adapters

Adapter for WebSphere MQ Integrator Broker ユーザーズ・ガイド

V 2.5.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、 89 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 Integrator バージョン 2.4.x、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： IBM WebSphere Business Integration Adapters
Adapter for WebSphere MQ Integrator Broker User Guide
V 2.5.x

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.1

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

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

目次

本書について

IBM(R) WebSphere(R) 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・プロセ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。

本書では、WebSphere MQ Integrator Broker 用のアダプターのインストール、構成、およびビジネス・オブジェクト開発について説明します。

対象読者

本書は、お客様のサイトで WebSphere Business Integration 製品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 Integrator (MQ Integrator Broker) 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。

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。

本書には、以下の 2 つの資料への参照が多数含まれています。「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)*」、および「*WebSphere InterChange Server* システム・インプリメンテーション・ガイド」本書を印刷する場合は、これらの資料も印刷すると便利です。

関連資料は以下のサイトからインストールできます。

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

  http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- アダプターを InterChange Server とともに使用する場合は、以下のサイトを参照してください。

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

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

- Message Brokers (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) の詳細については、以下のサイトを参照してください。
  http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- WebSphere Application Server の詳細については、以下のサイトを参照してください。http://www.ibm.com/software/webservers/appserv/library.html

これらのサイトでは、資料のダウンロード、インストール、および表示方法を簡単に説明しています。

表記上の規則

本書では、以下のような規則を使用しています。

`Courier フォント`	コマンド名、ファイル名、入力情報、システムが画面に出力した情報など、記述されたとおりの値を示します。
太字	初出語を示します。
イタリック、*イタリック*	変数名または相互参照を示します。
青のアウトライン	オンラインで表示したときにのみ見られる青のアウトラインは、相互参照用のハイパーリンクです。 アウトラインの内側をクリックすることにより、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、オプションのパラメーターです。
`...`	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、`option[,...]` は、複数のオプションをコンマで区切って指定できることを意味します。
`< >`	命名規則では、不等号括弧は名前の個々の要素を囲み、各要素を区別します (例: `<server_name><*connector_name*>tmp.log`)。
/, \	本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。UNIX システムの場合には、円記号をスラッシュ (/) 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*`%` および `$`*text*	パーセント (%) 符号で囲まれたテキストは、Windows の `text` システム変数またはユーザー変数の値を示します。UNIX 環境での同等の表記は `$`*text* であり、UNIX の *text* 環境変数の値を示します。
ProductDir	製品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 2.5.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 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 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。詳しくは、2 ページの『ブローカーの互換性』を参照してください。

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 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 の新機能

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

データ・ハンドラーを入力キューに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、30 ページの『データ・ハンドラーの InputQueues へのマッピング』を参照してください。

アダプターは、WebSphere MQ Integrator Broker と WebSphere InterChange Server の統合ブローカーのほかに、WebSphere MQ Event Broker をサポートするようになりました。

保証付きイベント・デリバリー機能が拡張されました。詳しくは、9 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 の新機能

InProgress キューは不要になったため、使用不可にすることができます。詳しくは、25 ページの『InProgressQueue』を参照してください。

コネクターは、MQSeries 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。詳しくは、3 ページの『アダプターの依存関係』を参照してください。

このリリースのコネクターには、ビジネス・オブジェクト処理のため UseDefaults プロパティーが用意されています。詳しくは、25 ページの『UseDefaults』を参照してください。

コネクターは、データ・ハンドラーがビジネス・オブジェクトに明示的にデフォルトの動詞を割り当てない場合、デフォルトの動詞を適用できるようになりました。詳しくは、23 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示されるようになりました。詳しくは、37 ページの『JMS ヘッダー、MQ Integrator Broker メッセージ・プロパティー、および動的子メタオブジェクト属性』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うほか、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。詳しくは、10 ページの『同期送達』を参照してください。

リリース 2.1.x の新機能

コネクターは国際化されました。詳しくは、15 ページの『ロケール依存データの処理』と 53 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。

このガイドでは、このアダプターを ICS と共に使用するための情報を提供します。

注: 保証イベント・デリバリー機能を使用するには、ICS のリリース 4.1.1.2 をインストールすることが必要です。

リリース 1.5.x の新機能

IBM WebSphere Business Integration Adapter for MQ Integrator には、MQ Integrator 用のコネクターが含まれます。このアダプターは、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す。統合ブローカーは、データ・ルーティングなどのサービスを提供します。アダプターには、以下の要素が含まれます。

- MQ Integrator に固有のアプリケーション・コンポーネント
- サンプル・ビジネス・オブジェクト
- IBM WebSphere Adapter Framework。コンポーネントは以下のとおりです。
  - コネクター・フレームワーク

– 開発ツール (Business Object Designer と IBM CrossWorlds System Manager を含む)
– API (CDK を含む)

本書では、このアダプターを ICS 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 ICS バージョン 4.1.1 を併用しないでください。

IBM CrossWorlds 4.1.x システムでコネクターが使用可能になりました。

リリース 1.4.x の新機能

本書の 1.4.x リリースには、欠陥を修正し IBM CrossWorlds インフラストラクチャー・リリース・バージョン 4.0.0 との互換性を保証するために、小さい変更が加えられています。

リリース 1.3.x の新機能

本書の 1.3.x リリースでは、以下の新機能と製品機能強化が追加されています。

- Create、Update、および Delete 操作を確認するための同期要求/応答処理のサポート。
- Retrieve、Retrieve By Content、および Exist 操作のサポート。
- 正常に処理されたメッセージ、アンサブスクライブされたメッセージ、およびエラーが発生したメッセージを含む、メッセージの全アーカイブ。
- 同じメッセージ・フォーマットを複数のビジネス・オブジェクトに割り当てる拡張機能。
- コネクターは、完全修飾 URI なしでローカル・キューを識別することが可能になりました。したがって、`InputQueueURI`、`InProgressQueueURI`、`UnsubscribedURI`、および `ErrorQueueURI` の各コネクター・プロパティーには、″URI″ サフィックスが含まれなくなりました。
- コネクター・メタオブジェクトにデフォルトの変換プロパティーが追加されました。したがって、コネクター・プロパティー `DefaultOutputQueueURI` が除去されました。

第 1 章 概要

MQ Integrator Broker 用のコネクターは、WebSphere Business Integration Adapter for WebSphere MQ Integrator Broker のランタイム・コンポーネントです。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。

コネクターは IBM の MQ Integrator Broker Version 2.2 製品と IBM WebSphere Business Integration システム・データ・ハンドラーを使用して、メッセージをビジネス・オブジェクトに変換し、ビジネス・オブジェクトをメッセージに変換します。つまり、コネクターは WebSphere Business Integration システムと IBM WebSphere MQ Integrater システムとを結ぶブリッジの役割をします。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 Adapters は、統合ブローカーと共に動作します。WebSphere MQ Integrator Broker 用コネクターは、以下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。

注: WebSphere MQ Integrator Broker バージョン 2.2 は、WebSphere MQ キュー間のメッセージを変換し、ルーティングするメッセージ・ブローカー製品です。このテクノロジーにより、アプリケーションは、リモート・キューとの間でメッセージを送受信して非同期的に通信することができます。MQ Integrator Broker での大きな変更点は、ユーザー定義のロジックに基づいてメッセージをフォーマット、格納、およびルーティングできる機能を追加するメッセージ・フローが追加されたことです。

Adapter for WebSphere MQ Integrator Broker 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。

- 『ブローカーの互換性』
- 3 ページの『アダプターのプラットフォーム』
- 3 ページの『アダプターの依存関係』
- 3 ページの『ロケール依存データ』

ブローカーの互換性

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

- アダプター・フレームワーク: WebSphere Business Integration Adapter Framework の以下のバージョン:
  - 2.1
  - 2.2
  - 2.3.x
  - 2.4
- 統合ブローカー:
  - WebSphere InterChange Server の以下のバージョン:
    - 4.11
    - 4.2
    - 4.2.1
    - 4.2.x
  - WebSphere Studio Application Developer Integration Edition バージョン 5.0.1 が組み込まれた WebSphere Application Server Enterprise バージョン 5.0.2

例外については、「リリース情報」を参照してください。

注: 統合ブローカーのインストールおよびその前提条件に関する説明については、以下の資料を参照してください。

WebSphere InterChange Server (ICS) については、「*システム・インストール・ガイド (UNIX 版)*」または「*システム・インストール・ガイド (Windows 版)*」を参照してください。

WebSphere Message Brokers (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) 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*」、および Message Broker のインストール関連資料を参照してください。これらの資料には、Web サイト http://www.ibm.com/software/integration/mqfamily/library/manualsa/ から検索できるものもあります。WebSphere Application Server については、「*アダプター実装ガイド (WebSphere Application Server)*」および http://www.ibm.com/software/webservers/appserv/library.html にある資料を参照してください。

アダプターのプラットフォーム

このアダプターは以下のプラットフォームでサポートされます。

- Windows 2000
- AIX 5.1、5.2
- Solaris 7、8
- HP-UX 11i

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 または 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。したがって、これらのいずれかのソフトウェア・リリースをインストールする必要があります。

  注: このアダプターは、WebSphere MQ 5.3 環境で SSL (Secure Socket Layer) をサポートしていません。アダプター・フレームワークと統合ブローカーの通信にとって適切な 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。
- さらに、IBM WebSphere MQ Java クライアント・ライブラリーも必要です。

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・テキストを送達できるように国際化されています。コネクターは、1 つの文字コー

1. ご使用の環境に「get 時の変換」方式の文字セット変換が実装されている場合、IBM から最新の MA88 (JMSクラス) をダウンロードする必要があります。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。

ドを使用する場所から別のコード・セットを使用する場所にデータを転送するとき、データの意味を保存するように文字変換を実行します。

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

エラー・メッセージと通知メッセージを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、53 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・アーキテクチャー

図 1 に、コネクター・コンポーネントと、WebSphere Business Integration システム内での各コンポーネントと MQSI メッセージ・ブローカーの関係を示します。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 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 です。

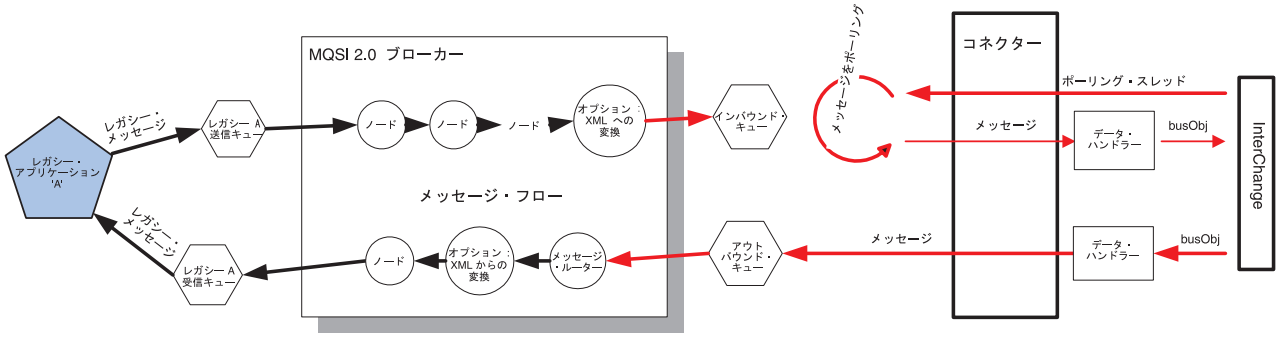

図 1. コネクターのアーキテクチャー

コネクターを使用すると、コラボレーションと、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。ただし、MQSI メッセージ・ブローカーは構文解析可能なすべてのメッセージをオプションで XML 形式に変換できるため、すべてのメッセージを XML 形式でデリバリーするようにコネクターを構成することを強くお勧めします。

これは、処理のために XML Data Handler を実装することを意味します。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。

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。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。

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・オブジェクト名とともに、メッセージがデータ・ハンドラーに渡されます。ビジネス・オブジェクトにメッセージの内容が正常に取り込まれると、コネクターはそのビジネス・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の間の通信方法

コネクターは、Java Message Service (JMS) の IBM による 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 2 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 MQSI に適したメッセージに変換し、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。

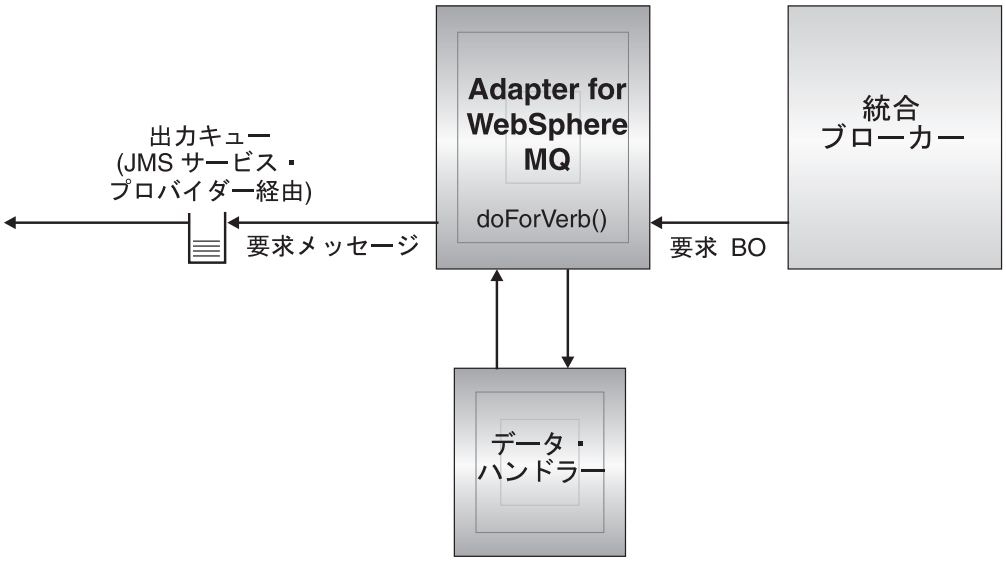

図 2. アプリケーションとコネクターの間の通信方法: メッセージ要求

メッセージ戻り

図 3 に、メッセージ戻りの方向を示します。`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、MQ Integrator Broker メタオブジェクトを使用して、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・ハンドラーがそのメッセージをビジネス・オブジェクトに変換します。設定される動詞には、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。

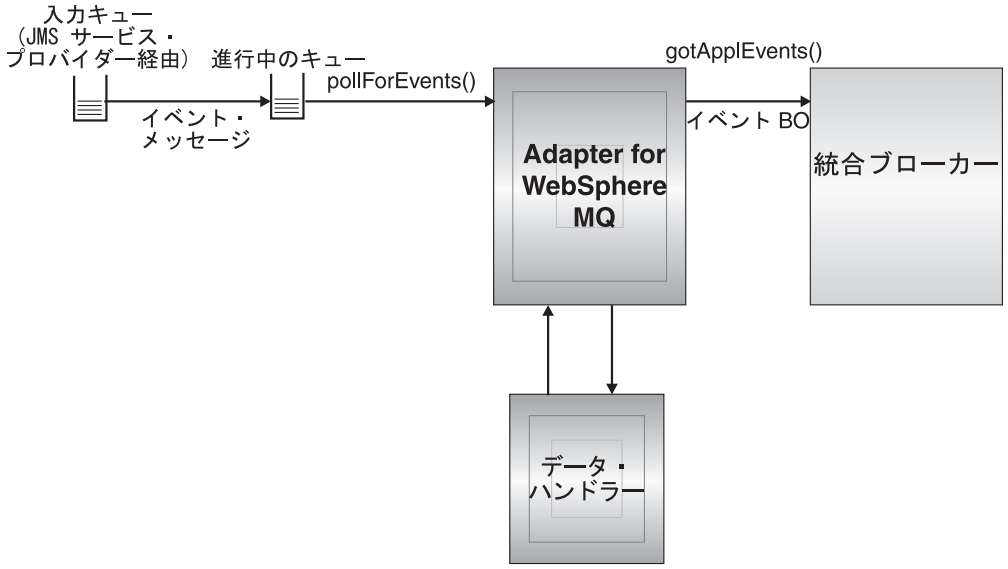

図 3. アプリケーションとコネクターの間の通信方法: メッセージ戻り

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。フォーマットがコネクター・メタオブジェクト内で定義されている場合、コネクターはデータ・ハンドラーを使用して適切な動詞付きのビジネス・オブジェクトを生成します。イベント失敗のシナリオについては、49 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 回デリバリーされるという可能性が少し残ります。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。メッセージは、処理が完了するまでキュー内に残ります。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ります。

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てから検索をコミットします。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。

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に実行中のキュー内で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に実行中のキュー内で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、初期化の際にコネクターが実行中のキュー内で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、初期化の際にコネクターが実行中のキュー内で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、49 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文とヘッダー (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・ストアから、JMS イベント・ストア、そして宛先の JMS キューまでの間で、イベントを失うことなく、また 2 度送りすることなく、確実に送達することができます。コネクターを JMS 対応にするには、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間の以降の通信は、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は確実に宛先に送達されます。このトランスポートの役割は、一度トランザクション・キュー・セッションが開始すると、メッセージはコミットが発行されるまで、確実にそのトランスポートのキャッシュに保持されます。障害が発生するか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: 保証イベント・デリバリー機能が使用されない場合、コネクターがイベントをパブリッシュしてから (コネクターがその pollForEvents() メソッドの内部から gotApplEvent() メソッドを呼び出してから) コネクターがイベント・レコードを削除することによりイベント・ストアを更新するまで (またはイベント・ストアを「イベント通知済み」状況に更新するまで)、障害を想定した小さなウィンドウが表示されます。このウィンドウで障害が発生すると、イベントは送信されますが、そのイベント・レコードは「進行中」の状況でイベント・ストアに残ります。コネクターが再始動すると、イベント・ストアにイベント・レコードが残っているため、これを送信する結果、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 対応コネクター用の保証付きイベント・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 ICS 統合ブローカーに送達できない場合、オブジェクトは FaultQueue (UnsubscribedQueue や ErrorQueue ではなく) に入り、状況表示と問題の記述を生成します。FaultQueue メッセージは、MQRFH2 フォーマットで書き込まれます。

ビジネス・オブジェクト要求

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、または Retrieve by Content 動詞を持つ ビジネス・オブジェクトの非同期送信をサポートしません。したがって、Retrieve 動詞、Exists 動詞、または Retrieve by Content 動詞のデフォルト・モードは同期送信です。

Create、Update、および Delete

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

非同期送達

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送信モードです。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た場合、コネクターは BON_SUCCESS を戻します。それ以外の場合は、BON_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。

同期送達

コネクターのプロパティーで `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。

MQ Integrator Broker の場合、コネクターは最初に、表 1 に示すヘッダーを持つメッセージを発行します。

表 1. 要求メッセージ記述子ヘッダー (MQMD)

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)
MsgType	メッセージ・タイプ	MQMT_DATAGRAM*
Report	要求されたレポート・メッセージのオプション	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*。
ReplyToQ	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*
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。

表 1 に示すヘッダーの後に、メッセージ本文が続きます。メッセージの本文は、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。

アプリケーションが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 2、3、および 4 に示すレポート・メッセージを発行します。

表 2. 応答メッセージ記述子ヘッダー (MQMD)

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
MsgType	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 3. 応答メッセージへの取り込み

動詞	Feedback フィールド	メッセージの本文
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 4. MQ Integrator Broker フィードバック・コードと WebSphere Business Integration システムの応答値

MQ Integrator フィードバック・コード	対応する WebSphere Business Integration システムの応答 *
MQFB_PAN または MQFB_APPL_FIRST	SUCCESS
MQFB_NAN または MQFB_APPL_FIRST + 1	FAIL
MQFB_APPL_FIRST + 2	VALCHANGE
MQFB_APPL_FIRST + 3	VALDUPES
MQFB_APPL_FIRST + 4	MULTIPLE_HITS
MQFB_APPL_FIRST + 5	FAIL_RETRIEVE_BY_CONTENT
MQFB_APPL_FIRST + 6	BO_DOES_NOT_EXIST
MQFB_APPL_FIRST + 7	UNABLE_TO_LOGIN (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)
MQFB_APPL_FIRST + 8	APP_RESPONSE_TIMEOUT

* 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。

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文に取り込みます。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・メッセージの本文にエラー・メッセージを含めることもできます。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、応答メッセージを取り出すと、デフォルトにより応答の 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、メッセージの本文にビジネス・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。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、メッセージの本文にビジネス・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 が要求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、メッセージの本文にエラー・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 JMS 機能は同期要求処理にのみ適用されます。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: 同期要求処理の場合にビジネス・オブジェクトを受信すると、コネクターは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るかどうかを検査します。response_selector が未定義の場合、コネクターは前述の相関 ID を使用して応答メッセージを識別します。

response_selector が定義されている場合、コネクターは以下の構文を持つ名前 - 値ペアを期待します。

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、値は以下に示すように単一引用符で囲む必要があります。

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

上記の例では、要求メッセージの発行後に、アダプターは ″Oshkosh″ と等しい correlationID 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、これを応答としてディスパッチします。

また、アダプターによる実行時置換を実行して、それぞれの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、整数を中括弧で囲んだ形式でプレースホルダーを指定することもできます。例えば、'{1}' のようになります。次に、後ろにコロンを付け、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、置換に使用する属性の索引の役割を持ちます。例えば、以下のメッセージ選択子の場合:

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

このメッセージ選択子は、アダプターに {1} を選択子の後ろの最初の属性の値 (この場合は、子オブジェクト MyDynamicMO の CorrelationId という属性) に置換するよう通知します。属性 CorrelationID の値が 123ABC であった場合、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および使用します。

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

これによって応答メッセージを識別します。

また、以下のように複数の置換対象を指定することもできます。

```
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 PrimaryId 属性の値に置換し、{2} を子コンテナー・オブジェクト Address の 5 番目の位置の AddressId の値に置換します。この方法を使用すると、応答メッセージ選択子内にあるビジネス・オブジェクトおよびメタオブジェクトのどの属性でも参照できます。Address[4].AddressId を使用した詳細検索の実行方法に関する詳細は、JCDK API のマニュアル (getAttribute メソッド) を参照してください。

以下のいずれかの状況が発生した場合、実行時にエラーが報告されます。

- '{}' シンボルの間に非整数を指定した場合
- 属性が定義されていない索引を指定した場合

- 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
- 属性パスの構文が誤っている場合

例えば、メッセージ選択子にリテラル値 '{' または '}' を組み込む場合は、それぞれ '{{' または "{}" を使用できます。また、これらの文字を属性値に組み込むこともできます。その場合、最初の "{" は不要です。エスケープ文字を使用した例を以下に示します。

```
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以下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特殊文字 ('{'、'}'、':'、または ';' など) を検出すると、これらの文字は照会ストリングに直接挿入されます。これにより、アプリケーション固有情報の区切り文字の役割も持つ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どのように抽出されるかを示します。

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

MyDynamicMO.CorrelationID に値 {A:B}C;D が含まれていた場合、コネクターはメッセージ選択子を以下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関する詳細は、JMS 1.0.1 仕様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 が必要です。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文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 5 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 5. 応答メッセージへの取り込み

動詞	Feedback フィールド	メッセージの本文
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。

表 5. 応答メッセージへの取り込み (続き)

動詞	Feedback フィールド	メッセージの本文
Exist	FAIL	(オプション) エラー・メッセージ。
	SUCCESS	

ロケール依存データの処理

コネクターは、2 バイト文字セットをサポートし、指定された言語でメッセージ・テキストを送達できるように国際化されています。コネクターは、1 つの文字コードを使用する場所から別のコード・セットを使用する場所にデータを転送するとき、データの意味を保存するように文字変換を実行します。

Java 仮想マシン (JVM) 内での Java ランタイム環境は、Unicode 文字コード・セットでデータを表します。Unicode には、ほとんどの既知の文字コード・セット (1 バイト系とマルチバイト系を含む) の文字に対応できるエンコード方式が組み込まれています。WebSphere Business Integration システムのほとんどのコンポーネントは Java で記述されています。したがって、ほとんどのインテグレーション・コンポーネントの間でデータが転送されても、文字変換の必要はありません。

エラー・メッセージと通知メッセージを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、53 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

第 2 章 コネクターのインストールおよび構成

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ともに動作させるための構成方法について説明します。

インストール作業の概要

MQ Integrator Broker 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、アダプター用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。『コネクターと関連ファイルのインストール』を参照してください。

コネクターと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については、「*WebSphere Business Integration Adapters インストール・ガイド*」を参照してください。この資料は、次の Web サイトの WebSphere Business Integration Adapters Infocenter にあります。

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

Windows のコネクター・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。

このユーティリティーは、コネクター・エージェントを *ProductDir*¥connectors¥WebSphereMQIntegratorBroker ディレクトリーにインストールし、そのコネクター・エージェントへのショートカットを「スタート」メニューに追加します。

表 6 で、コネクターにより使用される Windows ファイル構造について説明し、インストーラーによるコネクターのインストールを選択したときに自動的にインストールされるファイルを示します。

表 6. コネクター用にインストールされた Windows ファイル構造

***ProductDir* のサブディレクトリー**	**説明**
connectors¥WebSphereMQIntegratorBroker¥CWWebSphereMQIntegratorBroker.jar	WebSphere MQ Integrator コネクターによってのみ使用されるクラスを含みます。
connectors¥WebSphereMQIntegratorBroker¥start_WebSphereMQIntegratorBroker.bat	コネクター (Windows 2000) の始動スクリプト
connectors¥messages¥WebSphereMQIntegratorBrokerConnector.txt	コネクターのメッセージ・ファイル
repository¥WebSphereMQIntegratorBroker¥CN_WebSphereMQIntegratorBroker.txt	コネクターのリポジトリー定義。
connectors¥WebSphereMQIntegratorBroker¥samples¥LegacyItem¥LegacyItem.txt	サンプル・メッセージ
connectors¥WebSphereMQIntegratorBroker¥samples¥LegacyItem¥Sample_Integrator_Broker_Workspace.xml	サンプル・フローを含む MQ Integrator Broker ワークスペース
connectors¥WebSphereMQIntegratorBroker¥samples¥LegacyItem¥Sample_Event_Broker_Workspace.xml	サンプル・フローを含む MQ Event Broker ワークスペース
connectors¥WebSphereMQIntegratorBroker¥samples¥LegacyItem/WebSphereMQIntegratorBrokerConnector.cfg	コネクター構成ファイルのサンプル
connectors¥WebSphereMQIntegratorBroker¥amples¥LegacyItem¥PortConnector.cfg	ポート・コネクター構成ファイルのサンプル
connectors¥WebSphereMQIntegratorBroker¥samples¥LegacyItem¥Sample_WMQIB_LegacyItem.xsd	xml スキーマのサンプル
connectors¥WebSphereMQIntegratorBroker¥samples¥LegacyItem¥Sample_WMQIB_LegacyItem_XMLDoc.xsd	xml スキーマのサンプル
connectors¥WebSphereMQIntegratorBroker¥samples¥LegacyItem¥Sample_WMQIB_MO_Config.xsd	メタオブジェクトのサンプル
connectors¥WebSphereMQIntegratorBroker/samples/LegacyItem/Sample_WMQIB_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¥WebSphereMQIntegratorBroker¥amples¥LegacyItem¥Sample_WMQIB_MO_DataHandler_XMLConfig.xsd	XML データ・ハンドラー構成スキーマ・ファイルのサンプル

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 のコネクター・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。

このユーティリティーは、コネクター・エージェントを *ProductDir*/connectors/MQSIV2 ディレクトリーにインストールします。

表 7 では、コネクターにより使用される UNIX ファイル構造について説明し、インストーラーによるコネクターのインストールを選択したときに自動的にインストールされるファイルを示します。

表 7. コネクター用にインストールされた UNIX ファイル構造

***ProductDir* のサブディレクトリー**	**説明**
connectors/WebSphereMQIntegratorBroker/CWWebSphereMQIntegratorBroker.jar	WebSphere MQ Integrator コネクターによってのみ使用されるクラスを含みます。
connectors/WebSphereMQIntegratorBroker/start_WebSphereMQIntegratorBroker.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、インストーラーは、このコネクター・マネージャー・スクリプト用にカスタマイズされたラッパーを作成します。コネクターの始動と停止には、このカスタマイズ済みラッパーを使用してください。
connectors/messages/MQSIV2Connector.txt	コネクターのメッセージ・ファイル
repository/MQSIV2/CN_MQSIV2.txt	コネクターのリポジトリー定義。
connectors/WebSphereMQIntegratorBroker/samples/LegacyItem/LegacyItem.txt	サンプル・メッセージ
connectors/WebSphereMQIntegratorBroker/samples/LegacyItem/Sample_Integrator_Broker_Workspace.xml	サンプル・フローを含む MQ Integrator Broker ワークスペース
connectors/WebSphereMQIntegratorBroker/samples/LegacyItem/Sample_Event_Broker_Workspace.xml	サンプル・フローを含む MQ Event Broker ワークスペース
connectors/WebSphereMQIntegratorBroker/samples/LegacyItem/WebSphereMQIntegratorBrokerConnector.cfg	コネクター構成ファイルのサンプル
connectors/WebSphereMQIntegratorBroker/samples/LegacyItem/PortConnector.cfg	ポート・コネクター構成ファイルのサンプル
connectors/WebSphereMQIntegratorBroker/samples/LegacyItem/Sample_WMQIB_LegacyItem.xsd	xml スキーマのサンプル
connectors/WebSphereMQIntegratorBroker/samples/LegacyItem/Sample_WMQIB_LegacyItem_XMLDoc.xsd	xml スキーマのサンプル
connectors/WebSphereMQIntegratorBroker/samples/LegacyItem/Sample_WMQIB_MO_Config.xsd	メタオブジェクトのサンプル
connectors/WebSphereMQIntegratorBroker/samples/LegacyItem/Sample_WMQIB_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/WebSphereMQIntegratorBroker/samples/LegacyItem/Sample_WMQIB_MO_DataHandler_XMLConfig.xsd	XML データ・ハンドラー構成スキーマ・ファイルのサンプル

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。

メッセージ・フローの変更

各着信メッセージがデータ・ハンドラーに対応するフォーマットに変換されるように、メッセージ・フローを変更する必要があります。この変換は、メッセージがコネクターの入力キューに送信される前に行われる必要があります。

例えば、XML データ・ハンドラーを構成した場合、コネクターの入力キューにメッセージが送信される前に、メッセージが XML に変換されるようにメッセージ・フローを変更しておく必要があります。この変更 (XML データ・ハンドラーの場合) を行うには、メッセージ・フローの終わりに計算ノードを追加します。計算ノードには、図 4 に示す ESQL ステートメントが設定されていることが必要です。

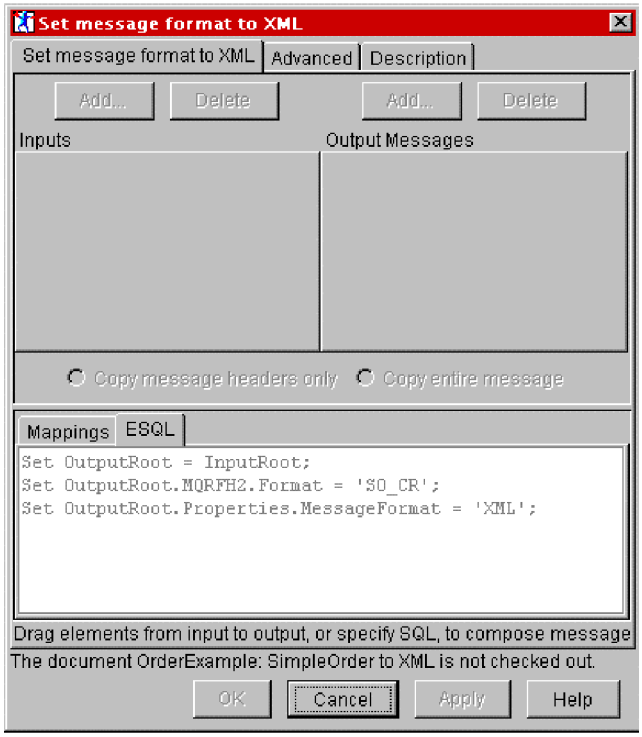

図 4. メッセージ・フォーマットを XML に設定

図 4 に、着信メッセージをコネクターが認識できるフォーマットに変換するように構成された計算ノードのサンプル・ビューを示します。この計算ノードが使用可能になった後は、オリジナルのメッセージを表す XML 文書がコネクターの入力キューに送信されます。

図 4 に示す ESQL テキスト領域内のフィールドは次のとおりです。

```
Set OutputRoot = InputRoot;
```

これにより、メッセージが出力のためにコピーされます。

```
Set OutputRoot.MQHRF2.Format = 'SO-CR';
```

これにより、コネクターがこのフォーマットをチェックして、メッセージを適切に変換することが保証されます。

```
SET OutputRoot.Properties.MessageFormat = 'XML';
```

これは、メッセージがデリバリー時に XML に変換される必要があることを MQ Integrator Broker に示します。

注: MQ Integrator Broker の Message Repository Manager (MRM) 内でカスタム・フォーマットを定義している場合は、メッセージ・フォーマットを XML に設定するだけで既存のフォーマットを XML に変換できます。このフォーマットは、MQHRF2 とは異なります。`OutputRoot.Properties.MessageFormat` は MRM に関するものですが、`OutputRoot.MQHRF2.Format` はメッセージを受信するアプリケーションのメッセージ・フォーマットを指定するために使用されます。

コネクターの構成

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一部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、 71 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 53 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について詳しくは、『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は、すべてのコネクターが使用する情報を提供します。これらのプロパティーの資料については、 53 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は、コネクター・エージェントが実行時に必要とする情報を提供します。また、コネクター固有の構成プロパティーを使用すると、コネクター・エージェントのコード変更や再ビルドを行わなくても、エージェント内の静的情報またはロジックを変更できます。

表 8 に、このコネクターに固有な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。

表 8. コネクター固有の構成プロパティー

名前	指定可能な値	デフォルト値	必須
ApplicationPassword	*ログイン・パスワード*		いいえ
ApplicationUserName	*ログイン・ユーザー ID*		いいえ
ArchiveQueue	*正常に処理されたメッセージのコピーが送信されるキュー*	MQWFCONN.ARCHIVE	いいえ
CCSID	キュー・マネージャー*接続用文字セット*	ヌル	はい
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	はい
DataHandlerConfigMO	データ・ハンドラー・メタオブジェクト	MO_DataHandler_Default	はい
DataHandlerMimeType	ファイルの *MIME タイプ*	text/xml	いいえ
DefaultVerb	コネクターがサポートしている任意の動詞。	Create	
ErrorQueue	*未処理のメッセージのキュー*	MQWFCONN.ERROR	いいえ
HostName	*WebSphere MQ サーバー*		いいえ
InDoubtEvents	FailOnStartup Reprocess IgnoreLogError	Reprocess	いいえ
InputQueue	ポーリング・キュー	MQWFCONN.IN	はい
InProgressQueue	*進行中イベント・キュー*	MQWFCONN.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*		いいえ
ReplyToQueue	コネクターからの要求発行時に応答メッセージがデリバリーされるキュー	MQWFCONN.REPLYTO	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	MQWFCONN.UNSUBSCRIBE	いいえ
UseDefaults	true または false	false	

ApplicationPassword

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

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 MQ Integrator Broker 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。*

ApplicationUserName

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

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 MQ Integrator Broker 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。*

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 = MQWFCONN.ARCHIVE

CCSID
キュー・マネージャー接続用文字セット。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していることが必要です。 27 ページの『キューの Uniform Resource Identifier (URI)』を参照してください。

デフォルト = ヌル

Channel
コネクターが MQ Integrator Broker 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 MQ Integrator Broker 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。*

ConfigurationMetaObject
コネクターの構成情報を含む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です。

デフォルト = MO_DataHandler_Default

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

デフォルト = text/xml

DefaultVerb
ポーリング時にデータ・ハンドラーによって動詞が設定されなかった場合、着信ビジネス・オブジェクトの内部に設定する動詞を指定します。

デフォルト= Create

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.QueueManager/ERROR

HostName
MQ Integrator Broker 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・ログを記録して即時にシャットダウンします。
- Reprocess: 残りのイベントを先に処理してから、入力キューのメッセージを処理します。
- Ignore: 進行中キューのすべてのメッセージを無視します。
- LogError: エラー・ログを記録しますが、シャットダウンはしません。

デフォルト = Reprocess

InputQueue
コネクターが新規メッセージをポーリングするメッセージ・キューです。コネクターは、セミコロンによって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 つのキューを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 にします。

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* の値が 2 で、MyQueueA 内に 2 つ、MyQueueB 内に 1 つ、MyQueueC 内に 5 つのメッセージがそれぞれ格納されている場合、コネクターは次のようにメッセージを検索します。

pollQuanity 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 サイクルのうちの 1 サイクル目) では、コネクターは MyQueueA、MyQueueB、および MyQueueC の各キューからそれぞれ 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 pollQuanity が 2 に設定されているため、コネクターは第 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 回ずつポーリングすると、メソッド pollForEvents への呼び出しは完了します。このメッセージ検索の順序を以下に示します。

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

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

デフォルト = queue://CrossWorlds.QueueManager/IN

InProgressQueue

処理中にメッセージが保持されるメッセージ・キューです。コネクターがこのキューなしで動作するように構成するには、System Manager を使用してコネクター固有のプロパティーからデフォルトの InProgressQueue 名を除去します。これを実行すると、始動時に、イベントの保留中にコネクターをシャットダウンするとイベント・デリバリーが正常に実行されない可能性があるという警告のプロンプトが出されます。

デフォルト = queue://CrossWorlds.QueueManaer/IN_PROGRESS

PollQuantity

pollForEvents スキャン中に 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

デフォルト = 1

Port

MQ Integrator Broker 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デリバリーされるキューです。

デフォルト = queue://CrossWorlds.QueueManager/REPLYTO

UnsubscribedQueue

サブスクライブされていないメッセージが送信されるキューです。

デフォルト = queue://CrossWorlds.QueueManager/UNSUBSCRIBED

注: *MQ Integrator Broker 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 操作において、UseDefaults に true が設定されている場合、コネクターは、各 isRequired ビジネス・オブジェクト属性に対して、デフォルト値または有効値が指定されているかどうかをチェックします。デフォルト値または有効値が指定されている場合は、Create 操作は成功します。このパラメーターに false が設定されている場合は、コネクターは、有効値のみをチェックします。有効値が指定されていない場合は、Create 操作は失敗します。デフォルト値は false です。

複数コネクター・インスタンスの作成

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

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

新規ディレクトリーの作成

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

```
ProductDir\connectors\connectorInstance
```

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

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

```
ProductDir\repository\connectorInstance
```

ビジネス・オブジェクト定義の作成

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

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

   ```
   ProductDir\repository\initialConnectorInstance
   ```

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

コネクター定義の作成

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

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

始動スクリプトの作成

始動スクリプトは以下のように作成します。

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

   dirname

2. この始動スクリプトを、 26 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. 始動スクリプトのショートカットを作成します (Windows のみ)。
4. 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。

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

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

キューの Uniform Resource Identifier (URI)

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

- キューが存在しているキュー・マネージャーの名前
- 別の `/`
- キューの名前
- (オプション) 残りのキュー・プロパティーの、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/IN?targetClient=1&persistence=5
```

表 9 に、キュー URI に対応するプロパティー名を示します。

表 9. MQ Integrator Broker キューの URI プロパティー

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。

表 9. *MQ Integrator Broker* キューの *URI* プロパティー *(続き)*

プロパティー名	説明	値
persistence	メッセージをディスクに「ハード化」するかどうか	1 = 非永続 2 = 永続 値 -1 は、このプロパティ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	宛先の文字セット	整数: MQ Integrator Broker の資料にリストされている有効な値。この値は、CCSID のコネクター固有構成プロパティーの値と一致していることが必要です。 23 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	MQ Integrator Broker の資料に記載されている整数値。

注: コネクターは、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、コネクターはデータ変換を JMS の IBM WebSphere MQ インプリメンテーションに依存します (IBM MQSeries 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 MQSeries または WebSphere MQ の 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。多くの場合、CCSID およびエンコードのサポートの欠如は、IBM の Web サイトから最新バージョンの IBM MQSeries Java クライアント・ライブラリーをダウンロードすることによって解決できます。それでも CCSID およびエンコードに関する問題が解消されない場合は、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。

メタオブジェクト属性の構成

MQ Integrator Broker 対応コネクターは、次の 2 種類のメタオブジェクトを認識および読み取りできます。

- 静的コネクター・メタオブジェクト
- 動的子メタオブジェクト

動的子メタオブジェクトの属性値は、静的メタオブジェクトの属性値を複写およびオーバーライドします。

静的メタオブジェクト

MQ Integrator Broker の構成メタオブジェクトは、さまざまなビジネス・オブジェクトに定義される変換プロパティーのリストで構成されます。ビジネス・オブジェクトの変換プロパティーを定義するには、最初にストリング属性を作成し、次に構文 `busObj_verb` を使用してそれに名前を付けます。例えば、動詞 Create 付きの Customer オブジェクトの変換プロパティーを定義するには、Customer_Create という名前の属性を作成します。属性のアプリケーション固有情報に、実際の変換プロパティーを指定します。

注: 静的メタオブジェクトが指定されていない場合、コネクターはポーリング時にメッセージ・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。そのような場合、コネクターは、ビジネス・オブジェクトを指定せずに、構成されているデータ・ハンドラーにメッセージ・テキストを渡します。データ・ハンドラーがテキストのみに基づいてビジネス・オブジェクトを作成することができない場合、コネクターはこのメッセージ・フォーマットを認識できないことを示すエラーを報告します。

表 10 に、メタオブジェクト・プロパティーを示します。

表 10. MQ Integrator Broker のメタオブジェクト・プロパティー

プロパティー名	説明
`InputFormat`	`InputFormat` は、特定のビジネス・オブジェクトに関連付けるメッセージ・フォーマットです。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。
`InputQueue`	新規メッセージを検出するためにコネクターがポーリングする入力キューです。 **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。データ・ハンドラーの InputQueues へのマッピングについては、 30 ページの『データ・ハンドラーの InputQueues へのマッピング』を参照してください。
`OutputFormat`	`OutputFormat` は、特定のビジネス・オブジェクトから作成されたメッセージに設定されます。`OutputFormat` が指定されていない場合は、入力フォーマットが (利用可能であれば) 使用されます。
`OutputQueue`	`OutputQueue` は、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出力キューです。
`ResponseTimeout`	応答を待機するときのタイムアウトまでの待機時間を、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`	このプロパティーが定義されていて値が `True` の場合、`ResponseTimeout` で指定された時間内に応答がないと、コネクターは APP_RESPONSE_TIMEOUT を戻します。応答メッセージを待機しているその他のすべてのスレッドは、InterChange Server に APP_RESPONSE_TIMEOUT を即時に戻します。これにより、InterChange Server はコネクターを終了します。

さらに、Default という名前の予約済みプロパティーもメタオブジェクト内で定義できます。このプロパティーが存在する場合は、このプロパティーのアプリケーション固有情報によってすべてのビジネス・オブジェクト変換プロパティーのデフォルト値が指定されます。

表 11. Customer_Create 用の MQ Integrator Broker 静的メタオブジェクトの構造

プロパティー名	アプリケーション固有のテキスト
Default	OutputFormat=CUST_OUT; OutputQueue=QueueA; ResponseTimeout=10000; TimeoutFatal=False

アプリケーション固有の情報

アプリケーション固有情報のフォーマットは、名前と値の各ペアをセミコロンで区切ったストリングです。例えば、次のようにします。

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

データ・ハンドラーの InputQueues へのマッピング

静的メタオブジェクトのアプリケーション固有情報内の InputQueue プロパティーを使用して、データ・ハンドラーを入力キューに関連付けることができます。この機能は、フォーマットや型変換の要件が異なる複数の取引先との取引を行なう際に役立ちます。これを行なうには、次の手順を実行する必要があります。

1. コネクター固有のプロパティー (24 ページの『InputQueue』を参照) を使用して、1 つ以上の入力キューを構成する。
2. 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内の属性は、データ・ハンドラーを CompReceipts という名前の InputQueue に関連付けます。

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

入力フォーマットの多重定義

コネクターは通常、メッセージを検索するときに、ビジネス・オブジェクトと動詞の特定の 1 つの組み合わせと入力フォーマットを突き合わせます。そして、コネクターはビジネス・オブジェクト名とメッセージの内容をデータ・ハンドラーに渡します。これにより、データ・ハンドラーは、メッセージの内容がユーザーの予期するビジネス・オブジェクトに対応していることを確認できます。

しかし、複数のビジネス・オブジェクトに対して同じ入力フォーマットが定義されている場合、コネクターは、データ・ハンドラーにデータを渡す前に、データがどのビジネス・オブジェクトを表すのかを判別できません。そのような場合、コネクターはデータ・ハンドラーにメッセージの内容のみを渡し、生成されるビジネス・オブジェクトに基づいて変換プロパティーを調べます。したがって、データ・ハンドラーは、メッセージの内容のみに基づいてビジネス・オブジェクトを決定する必要があります。

生成されたビジネス・オブジェクトに動詞が設定されていない場合、コネクターはビジネス・オブジェクトの動詞の種類を限定せずに、このビジネス・オブジェクトに定義されている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は、コネクターは特定の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。

サンプル・メタオブジェクト

以下に示す静的メタオブジェクトは、動詞 Create、Update、Delete、および Retrieve を使用する Customer ビジネス・オブジェクトを変換するコネクターを構成します。このメタオブジェクトには、属性 `Default` が定義されています。したがって、コネクターはこの属性の変換プロパティー `(OutputQueue=CustomerQueue1;ResponseTimeout=5000;TimeoutFatal=true)` を使用します。そのため、属性によって別のものが指定されない限りは、コネクターはすべてのビジネス・オブジェクトをキュー `CustomerQueue1` に送信し、応答メッセージを待ちます。5000 ミリ秒以内に応答がなければ、コネクターは即時に終了します。

動詞 Create 付き Customer オブジェクト: 属性 `Customer_Create` は、フォーマットが `NEW` であるすべてのメッセージを動詞 Create 付きの Customer ビジネス・オブジェクトに変換する必要があることをコネクターに示します。出力フォーマットは定義されていないため、コネクターは、入力用に定義されたフォーマット (この場合は `NEW`) を使用して、このオブジェクトと動詞の組み合わせを表すメッセージを送信します。

動詞 Update および Delete 付き Customer オブジェクト: 入力フォーマット `MODIFY` は多重定義されています。入力フォーマットは、動詞 Update 付きのビジネス・オブジェクト Customer と動詞 Delete 付きのビジネス・オブジェクト Customer の両方に対して定義されています。検索されたこのフォーマットのメッセージを正常に処理するためには、データ・ハンドラーが判別できるように、メッセージの内容にビジネス・オブジェクトと動詞が含まれている必要があります (30 ページの『入力フォーマットの多重定義』を参照してください)。要求処理操作では、出力フォーマットが定義されていないため、コネクターは入力フォーマット `MODIFY` を使用していずれかの動詞のメッセージを送信します。

動詞 Retrieve 付き Customer オブジェクト: 属性 `Customer_Retrieve` は、動詞 Retrieve 付きの Customer タイプのビジネス・オブジェクトを、フォーマット Retrieve のメッセージとして送信する必要があることを指定します。タイムアウトまでにコネクターが最長 10000 ミリ秒まで待機するよう、デフォルトの応答時間がオーバーライドされています (応答がなければ、コネクターは終了します)。

```
[ReposCopy]
Version = 3.1.0
Repositories = 1cHyILNuPTc=
```

```
[End]
[BusinessObjectDefinition]
Name = Sample_MO
Version = 1.0.0

[Attribute]
Name = Default
Type = String
Cardinality = 1
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = OutputQueue=CustomerQueue1;ResponseTimeout=5000;TimeoutFatal=true
IsRequiredServerBound = false
[End]
[Attribute]
Name = Customer_Create
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = InputFormat=NEW
IsRequiredServerBound = false
[End]
[Attribute]
Name = Customer_Update
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = InputFormat=MODIFY
IsRequiredServerBound = false
[End]
[Attribute]
Name = Customer_Delete
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = InputFormat=MODIFY
IsRequiredServerBound = false
[End]
[Attribute]
Name = Customer_Retrieve
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = OutputFormat=RETRIEVE;ResponseTimeout=10000
IsRequiredServerBound = false
[End]
[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
```

```
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

動的子メタオブジェクト

静的メタオブジェクトに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、オプションで、コネクターが実行時にビジネス・オブジェクト・インスタンスごとに指定されたメタデータを受け入れることができます。

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。動的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メタオブジェクトによって指定できる変換プロパティーを複写します。

動的子メタオブジェクト・プロパティーは静的メタオブジェクト・プロパティーの値をオーバーライドするため、動的子メタオブジェクトを指定する場合は、静的メタオブジェクトを指定するコネクター・プロパティーを含める必要はありません。したがって、動的子メタオブジェクトは静的メタオブジェクトから独立して使用でき、また、静的メタオブジェクトは動的子メタオブジェクトから独立して使用できます。

注: コネクターは、同期イベント・デリバリー中にコラボレーション名を提供するための動的子メタオブジェクトの使用をサポートしません。

表 11 および 12 に、ビジネス・オブジェクト Customer_Create に対応する静的子メタオブジェクトと動的子メタオブジェクトのサンプルをそれぞれ示します。アプリケーション固有情報は、名前と値のペアをセミコロンで区切ったものとして構成されます。

表 12. Customer_Create 用の MQ Integrator Broker 動的子メタオブジェクトの構造

プロパティー名	値
OutputFormat	CUST_OUT
OutputQueue	QueueA
ResponseTimeout	10000
TimeoutFatal	False

コネクターは、受け取ったトップレベルのビジネス・オブジェクトのアプリケーション固有のテキストをチェックし、タグ cw_mo_conn が子メタオブジェクトを指定

しているかどうかを調べます。このタグが子メタオブジェクトを指定している場合、動的子メタオブジェクトの属性値は静的メタオブジェクトの属性値をオーバーライドします。

ポーリング時の動的子メタオブジェクトの取り込み

ポーリング時に検索されたメッセージに関してより多くの情報を持つコラボレーションを提供するために、コネクターは動的メタオブジェクトの特定の属性を (作成されるビジネス・オブジェクトにすでに定義されていれば) 取り込みます。

表 13 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 13. ポーリング用の MQ Integrator Broker 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 13 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。

シナリオの例:

- コネクターは、フォーマット CUST_IN のメッセージをキュー MyInputQueue から検索します。
- コネクターはこのメッセージを Customer ビジネス・オブジェクトに変換し、アプリケーション固有のテキストをチェックして、メタオブジェクトが定義されているかどうかを判別します。
- メタオブジェクトが定義されている場合、コネクターはそのメタオブジェクトのインスタンスを作成し、それに従って InputQueue 属性と InputFormat 属性を取り込み、利用可能なコラボレーションにビジネス・オブジェクトを発行します。

動的子メタオブジェクトのサンプル

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

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

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

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
```

```
[End]

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

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

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]
```

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

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

JMS ヘッダー、MQ Integrator Broker メッセージ・プロパティー、および動的子メタオブジェクト属性

動的メタオブジェクトに属性を追加すると、メッセージ・トランスポートに関する詳細情報を取得したり、メッセージ・トランスポートの管理を強化したりできます。このような属性を追加することで、JMS プロパティーの変更、要求ごとの ReplyToQueue の制御 (アダプターのプロパティーに指定されたデフォルトの ReplyToQueue を使用しない)、およびメッセージ CorrelationID のターゲット変更が可能になります。このセクションでは、これらの属性、および同期モードと非同期モードにおいて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。

以下の属性は JMS および MQ Integrator Broker のヘッダー・プロパティーを反映し、動的メタオブジェクトで認識されます。

表 14. 動的メタオブジェクトのヘッダー属性

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
優先順位	読み取り/書き込み	JMSPriority
Destination	読み取り	JMSDestination
Expiration	読み取り	JMSExpiration
MessageID	読み取り	JMSMessageID
Redelivered	読み取り	JMSRedelivered
TimeStamp	読み取り	JMSTimeStamp
型	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もまた、要求処理中に応答メッセージが発行されると動的メタオブジェクトに取り込まれます。読み取り/

書き込み属性は、要求処理中に作成されたメッセージ・ヘッダーに設定されます。イベント通知中に、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て動的メタオブジェクトに取り込まれます。

これらの属性の解釈と使用法については、以下のセクションで説明します。

注: いずれの属性も必須ではありません。独自のビジネス・プロセスに関連した動的メタオブジェクトに任意の属性を追加することができます。

同期イベント通知: 同期イベント処理の場合、アダプターはイベントを送付して統合ブローカーからの応答を待機してから、応答メッセージをアプリケーションに通知します。ビジネス・データに対する変更はすべて戻された応答メッセージに反映されます。イベントを送付する前に、アダプターは非同期イベント通知の場合と同様に動的メタオブジェクトへの取り込みを実行します。動的メタオブジェクトに設定された値は、以下に示すように応答発行ヘッダーに反映されます (動的メタオブジェクト内のその他の読み取り専用ヘッダー属性はすべて無視されます)。

- **CorrelationID** 動的メタオブジェクトに CorrelationId 属性が含まれている場合は、発信元アプリケーションによって想定される値に設定する必要があります。アプリケーションは CorrelationID を使用して、コネクターから戻されたメッセージを元の要求と突き合わせます。CorrelationID に予期しない値や無効値を設定すると、問題が発生します。この属性を使用する前に、アプリケーションで相関要求および応答メッセージがどのように処理されるかを判別しておくと役に立ちます。同期要求で CorrelationID への取り込みを実行するには 4 つのオプションがあります。
  1. 値を未変更のままにする。応答メッセージの CorrelationID を要求メッセージの CorrelationID と同じにする。これは、WebSphere MQ オプション MQRO_PASS_CORREL_ID と同じです。
  2. 値を CxIgnore に変更する。デフォルトでは、コネクターは要求のメッセージ ID を応答の CorrelationID にコピーします。これは、WebSphere MQ オプション MQRO_COPY_MSG_ID_TO_CORREL_ID と同じです。
  3. 値を CxBlank に変更する。コネクターは応答メッセージに CorrelationID を設定しません。
  4. 値をカスタム値に変更する。これには、応答を処理するアプリケーションがカスタム値を認識する必要があります。

  属性 CorrelationID をメタオブジェクト内に指定しない場合、コネクターは CorrelationID を自動的に処理します。
- **ReplyToQueue** 属性 ReplyToQueue に別のキューを指定して動的メタオブジェクトを更新した場合、コネクターは指定したキューに応答メッセージを送信します。この方法はお勧めできません。コネクターが別のキューに応答メッセージを送信するように構成すると、通信に障害が発生する可能性があります。これは、要求メッセージに特定の応答キューを設定したアプリケーションは、そのキューで応答を待機すると想定されるためです。
- **JMS properties** 更新済みビジネス・オブジェクトがコネクターに戻されるときに、動的メタオブジェクト内で JMS Properties 属性に設定された値が応答メッセージに設定されます。

非同期要求の処理: コネクターは、動的メタオブジェクトがある場合にはこれを使用して、要求メッセージを発行する前に取り込みを実行します。コネクターは要求メッセージを送信する前に以下のステップを実行します。

1. 属性 CorrelationID が動的メタオブジェクト内にある場合、コネクターはアウトバウンド要求メッセージの CorrelationID をこの値に設定します。
2. 属性 ReplyToQueue が動的メタオブジェクトに指定されている場合、コネクターは要求メッセージを介してこのキューを渡し、このキューで応答を待ちます。これによってコネクター構成プロパティーに指定した ReplyToQueue 値をオーバーライドできます。さらに、負の ResponseTimeout を指定した場合 (つまり、コネクターが応答を待機しないよう指定した場合) は、コネクターが実際には応答を待機しない場合でも ReplyToQueue が応答メッセージに設定されます。
3. 属性 DeliveryMode を 2 に設定すると、メッセージは永続的に送信されます。DeliveryMode を 1 に設定すると、メッセージは永続的に送信されません。その他の値を設定すると、コネクターに障害が発生します。MO に DeliveryMode を指定しないと、JMS プロバイダーが永続設定を確立します。
4. 属性 Priority を指定すると、コネクターが発信要求に値を設定します。Priority 属性には 0 から 9 までの値を設定できます。その他の値を指定すると、コネクターが終了します。
5. 属性 JMSProperties が動的メタオブジェクトに指定されている場合、子動的メタオブジェクトに指定された対応する JMS プロパティーは、コネクターによって送信されたアウトバウンド・メッセージに設定されます。

注: 動的メタオブジェクト内のヘッダー属性が未定義の場合、または CxIgnore が指定されている場合、コネクターはデフォルトの設定値を使用します。

同期要求の処理: コネクターは、動的メタオブジェクトがある場合にはこれを使用して、要求メッセージを発行する前に取り込みを実行します。動的メタオブジェクトにヘッダー属性が含まれている場合、コネクターは応答メッセージ内にある対応する新規の値をこの属性に取り込みます。コネクターは応答メッセージの受信後に (メタオブジェクトへのトランスポート関連データの取り込み以外に) 以下のステップを実行します。

1. 属性 CorrelationID が動的メタオブジェクト内にある場合、アダプターはこの属性を応答メッセージに指定された JMSCorrelationID で更新します。
2. 属性 ReplyToQueue が動的メタオブジェクト内に定義されている場合、アダプターはこの属性を応答メッセージに指定された JMSReplyTo の名前で更新します。
3. 属性 DeliveryMode が動的メタオブジェクトに含まれている場合、アダプターはこの属性をメッセージの JMSDeliveryMode ヘッダー・フィールドの値で更新します。
4. 属性 Priority が動的メタオブジェクトに含まれている場合、アダプターはこの属性をメッセージの JMSPriority ヘッダー・フィールドの値で更新します。
5. 属性 Destination が動的メタオブジェクト内に定義されている場合、アダプターはこの属性を応答メッセージに指定された JMSDestination の名前で更新します。

6. 属性 Expiration が動的メタオブジェクトに含まれている場合、アダプターはこの属性をメッセージの JMSExpiration ヘッダー・フィールドの値で更新します。
7. 属性 MessageID が動的メタオブジェクトに含まれている場合、アダプターはこの属性をメッセージの JMSMessageID ヘッダー・フィールドの値で更新します。
8. 属性 Redelivered が動的メタオブジェクトに含まれている場合、アダプターはこの属性をメッセージの JMSRedelivered ヘッダー・フィールドの値で更新します。
9. 属性 TimeStamp が動的メタオブジェクトに含まれている場合、アダプターはこの属性をメッセージの JMSTimeStamp ヘッダー・フィールドで更新します。
10. 属性 Type が動的メタオブジェクトに含まれている場合、アダプターはこの属性をメッセージの JMSType ヘッダー・フィールドの値で更新します。
11. 属性 UserID が動的メタオブジェクトに含まれている場合、アダプターはこの属性をメッセージの JMSXUserID ヘッダー・フィールドの値で更新します。
12. 属性 AppID が動的メタオブジェクトに含まれている場合、アダプターはこの属性をメッセージの JMSXAppID プロパティー・フィールドの値で更新します。
13. 属性 DeliveryCount が動的メタオブジェクトに含まれている場合、アダプターはこの属性をメッセージの JMSXDeliveryCount ヘッダー・フィールドの値で更新します。
14. 属性 GroupID が動的メタオブジェクトに含まれている場合、アダプターはこの属性をメッセージの JMSXGroupID ヘッダー・フィールドの値で更新します。
15. 属性 GroupSeq が動的メタオブジェクトに含まれている場合、アダプターはこの属性をメッセージの JMSXGroupSeq ヘッダー・フィールドの値で更新します。
16. 属性 JMSProperties が動的メタオブジェクト内に定義されている場合、アダプターは子オブジェクトに定義されたすべてのプロパティーを、応答メッセージで検出した値で更新します。子オブジェクトに定義されたプロパティーがメッセージ内に存在しない場合は、値は CxBlank に設定されます。

注: 動的メタオブジェクトを使用して要求メッセージに設定された CorrelationID を変更しても、アダプターが応答メッセージを識別する方法に影響はありません。デフォルトでは、アダプターは、任意の応答メッセージの CorrelationID がアダプターから送られた要求のメッセージ ID と等しいと仮定します。

エラー処理: メッセージに対して JMS プロパティーの読み取りまたは書き込みを実行できない場合、コネクターはエラーをログに記録し、要求またはイベントは失敗します。ユーザー指定の ReplyToQueue が存在しない場合またはアクセスできない場合は、コネクターはエラーをログに記録し、要求は失敗します。CorrelationID が無効であるか設定できない場合、コネクターはエラーをログに記録し、要求は失敗します。いずれの場合も、ログに記録されるメッセージはコネクター・メッセージ・ファイルから読み込まれます。

始動ファイルの構成

WebSphere MQ Integrator 対応コネクターを開始する前に、始動ファイルを構成しておく必要があります。

Windows

Windows プラットフォーム用のコネクターの構成を完了するためには、次のようにして start_WebSphereMQIntegratorBroker.bat ファイルを変更する必要があります。

1. start_WebSphereMQIntegratorBroker.bat ファイルを開きます。
2. スクロールして「Set the directory containing your WebSphereMQ 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_WebSphereMQIntegratorBroker.sh ファイルを変更する必要があります。

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

ProductDir\connectors*connName*

ここで、*connName* はコネクターを示します。始動スクリプトの名前は、表 15 に示すように、オペレーティング・システム・プラットフォームによって異なります。

表 15. コネクターの始動スクリプト

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

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

- Windows システムで**「スタート」**メニューから。

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

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

– UNIX ベースのシステム:

```
connector_manager_connName -start
```

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

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

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

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

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

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

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

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

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

コネクターの停止

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

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

  – Windows システムでは、始動スクリプトを起動すると、そのコネクター用の別個の「コンソール」ウィンドウが作成されます。このウィンドウで、「Q」と入力して Enter キーを押すと、コネクターが停止します。

– UNIX ベースのシステムでは、コネクターはバックグラウンドで実行されるため、別ウィンドウはありません。代わりに、次のコマンドを実行してコネクターを停止します。

```
connector_manager_connName -stop
```

ここで、*connName* はコネクターの名前です。

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

  このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。

- System Monitor から (WebSphere InterChange Server 製品のみ)

  このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。

- Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムのシャットダウン時に、コネクターは停止します。

第 3 章 ビジネス・オブジェクトの作成または変更

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。ビジネス・オブジェクトでは、メタデータはアプリケーションについてのデータです。メタデータは、ビジネス・オブジェクト定義内に格納され、コネクターがアプリケーションと対話するために役立ちます。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。実際、コネクターは、MQ Integrator Broker アプリケーションのいかなる変更によっても影響を受けない必要があります。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を前提とします。したがって、MQ Integrator Broker 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・オブジェクトを実装する際のガイドとして利用できます。

コネクターのビジネス・オブジェクトの構造

コネクターをインストールした後で、MQ Integrator Broker のビジネス・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。命名規則については、「*Naming IBM WebSphere InterChange Server Components*」を参照してください。

コネク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 = Example_SimpleOrder
Version = 1.0.0
AppSpecificInfo = m_SimpleOrder

[Attribute]
Name = ProductCode
Type = String
Cardinality = 1
MaxLength = 3
IsKey = true
IsForeignKey = false
IsRequired = true
AppSpecificInfo = e_ProductCode;type=pcdata;
DefaultValue = W02
IsRequiredServerBound = false
[End]

[Attribute]
Name = ProductDescription
Type = String
Cardinality = 1
MaxLength = 20
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = e_ProductDescription;type=pcdata;
DefaultValue = Deluxe Widget
IsRequiredServerBound = false
[End]
[Attribute]
Name = ProductQuantity
Type = Integer
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = true
AppSpecificInfo = e_ProductQuantity;type=pcdata;
DefaultValue = 1
IsRequiredServerBound = false
[End]

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

```
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

[BusinessObjectDefinition]
Name = Example_SimpleOrder_MRM
Version = 1.0.0
AppSpecificInfo = MRM

[Attribute]
Name = xmlns
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = xmlns;type=attribute
DefaultValue = www.mrmnames.net/DIN94VC078001
IsRequiredServerBound = false
[End]
[Attribute]
Name = SimpleOrder
Type = Example_SimpleOrder
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = m_SimpleOrder
IsRequiredServerBound = false
[End]
[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
```

```
[End]

[Verb]
Name = Retrieve
[End]

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

[BusinessObjectDefinition]
Name = Example_SimpleOrder_XMLDoc
Version = 1.0.0

[Attribute]
Name = XMLDeclaration
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = type=pi;
DefaultValue = xml version="1.0"
IsRequiredServerBound = false
[End]
[Attribute]
Name = Doctype
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = type=doctype;
DefaultValue = DOCTYPE MRM PUBLIC "www.mrmnames.net/DIN94VC078001" "DIN94VC078001"
IsRequiredServerBound = false
[End]

[Attribute]
Name = MRM_Wrapper
Type = Example_SimpleOrder_MRM
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = true
AppSpecificInfo = MRM
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]

[Verb]
```

```
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

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、MQSIV2Connector.txt という名前のメッセージ・ファイルに格納されます。(このファイルの名前は、コネクター構成標準プロパティー 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 ABON_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 APP_RESPONSE_TIMEOUT または UNABLE_TO_LOGIN の応答メッセージを受信しました。

アンサブスクライブされたビジネス・オブジェクト

コネクターは、次の場合に、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。

- 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コネクターが検索した場合、または gotApplEvent() メソッドによって NO_SUBSCRIPTION_FOUND コードが戻された場合。
- コネクターがメッセージを検索したが、FORMAT フィールドのテキストをビジネス・オブジェクト名に関連付けることができない場合。

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。

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 APP_RESPONSE_TIMEOUT コードを戻し、イベントは InProgress キューに置かれたままになります。

データ・ハンドラーによる変換

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、BON_FAIL が戻されます。

トレース

トレースはオプションのデバッグ機能で、オンにするとコネクタ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、*第 2 章* のコネクター構成プロパティーについて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 (Java 用)*」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 に通知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　データ形式 (例えば XML) からビジネス・オブジェクトへの変換およびビジネス・オブジェクトからデータ形式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。

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 MQSeries 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 `MQSERIES_JAVA_LIB` が IBM MQSeries 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 MQSeries 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 `MQSERIES_JAVA_LIB` が IBM MQSeries 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 MQSeries 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 `MQSERIES_JAVA_LIB` が IBM MQSeries 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 MQSeries 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` [Windows] または `libmqjbnd01.so` [UNIX]) を見つけることができません。パスに IBM MQSeries 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。詳細は、17 ページの『第 2 章 コネク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。
pollForEvents 時に、次の JMS 例外が報告された後でコネクターがシャットダウンしました: MQJMS1000: Failed to create JMS message	このエラーは MQSeries Java API に関するエラーであり、コネクター自身のエラーではありません。このエラーは、多くの場合、WebSphere MQ Integrator Broker 製品自体と同じマシン上でコネクターを実行しているときに発生します。この問題を解決するには、以下の手順を実行する必要があります。 1. ファイル %MQSIV2_INSTALL_FOLDER%\Dependencies\JRE_122_4.zip をフォルダー %CROSSWORLDS%\connectors\MQSIV2\Dependencies\jre_122_Rel4 に unzip します。 2. %CROSSWORLDS%¥connectors¥MQSIV2¥start_MQSIV2.bat を開き、次の 2 つの行のコメントをはずします。oset PATH=%CONNDIR%¥Dependencies¥jre_122_Rel4¥bin...oset JAVA=%CONNDIR%¥Dependencies¥jre_122_Rel4¥lib¥rt.jar 3. コネクターを再始動します。
pollForEvents 時に、次の JMS 例外が報告された後でコネクターがシャットダウンしました: MQJMS1052: Unrecognised [sic] JMS Message class	このエラーは、JMS 準拠のアプリケーションから発生したメッセージを WebSphere MQ Integrator Broker が変更したことによって発生することがあります。このエラーを訂正するには、WebSphere MQ Integrator Broker 計算ノードに次の SQL ステートメントを追加することにより、問題が発生したメッセージから残りの JMS 情報を除去します: SET OutputRoot.MQRFH2.jms = null;

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

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

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

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

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

新規プロパティーと削除されたプロパティー

以下の標準プロパティーは、本リリースで追加されました。

新規プロパティー

- XMLNameSpaceFormat

削除されたプロパティー

- RestartCount
- RHF2MessageDomain

標準コネクター・プロパティーの構成

アダプター・コネクターには 2 種類の構成プロパティーがあります。

- 標準構成プロパティー
- コネクター固有の構成プロパティー

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

Connector Configurator の使用

Connector Configurator からコネクター・プロパティーを構成します。 Connector Configurator には、System Manager からアクセスします。 Connector Configurator の使用方法の詳細は、Connector Configurator に関する付録を参照してください。

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

プロパティー値の設定と更新

プロパティーのフィールド長のデフォルトは 255 文字です。

コネクターは、以下の順序に従ってプロパティーの値を決定します (最も番号の大きい項目は、他の項目の値をオーバーライドします)。

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

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

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

特定のプロパティ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、または次に示すプロパティーの要約の表の**「更新メソッド」**列を参照してください。

標準プロパティーの要約

表 16 は、標準コネクター構成プロパティーの早見表です。標準プロパティーの依存関係は RepositoryDirectory に基づいているため、コネクターによっては使用されないプロパティーがあり、使用する統合ブローカーによってプロパティーの設定が異なる可能性があります。

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

表 16. 標準構成プロパティーの要約

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

表 16. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
DeliveryTransport	MQ、IDL、または JMS	JMS	コンポーネント再始動	Repository Directory がローカルの場合は、値は JMS のみ
DuplicateEventElimination	True または False	False	コンポーネント再始動	JMS トランスポートのみ、Container Managed Events は <NONE> でなければならない
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory または CxCommon.Messaging.jms.SonicMQFactory または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	FactoryClassName が IBM の場合は crossworlds.queue.manager を使用。FactoryClassName が Sonicの場合 localhost:2506 を使用。	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMaxNativeStackSize	スタックのサイズ (キロバイト単位)	128k	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	Repository Directory は <REMOTE>
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならない

表 16. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
Locale	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR **注:** これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	True または False	False	コンポーネント再始動	Repository Directory は <REMOTE> でなければならない
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならない
MessageFileName	パスまたはファイル名	InterchangeSystem.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	*CONNECTORNAME*/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は True でなければならない
OADAutoRestartAgent	True または False	False	動的	Repository Directory は <REMOTE> でなければならない
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければならない
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければならない
PollEndTime	HH:MM	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒) no (ポーリングを使用不可にする) key (コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力された場合にのみポーリングする)	10000	動的	

表 16. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
PollQuantity	1 から 500	1	エージェント再始動	JMS トランスポートのみ: Container Managed Events を指定
PollStartTime	HH:MM (HH は 0 から 23、MM は 0 から 59)	HH:MM	コンポーネント再始動	
RepositoryDirectory	メタデータ・リポジトリーの場所		エージェント再始動	ICS の場合は <REMOTE> に設定する。WebSphere MQ Message Brokers および WAS の場合: C:¥crossworlds¥repository に設定する
RequestQueue	有効な JMS キュー名	*CONNECTORNAME*/REQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
ResponseQueue	有効な JMS キュー名	*CONNECTORNAME*/RESPONSEQUEUE	コンポーネント再始動	Delivery Transport が JMS の場合: Repository Directory が <REMOTE> の場合のみ必要
RestartRetryCount	0 から 99	3	動的	
RestartRetryInterval	適切な正数 (単位: 分): 1 から 2147483547	1	動的	
SourceQueue	有効な WebSphere MQ 名	*CONNECTORNAME*/SOURCEQUEUE	エージェント再始動	Delivery Transport が JMS であり、かつ Container Managed Events が指定されている場合のみ
SynchronousRequestQueue		*CONNECTORNAME*/SYNCHRONOUSREQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
SynchronousRequestTimeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	Delivery Transport は JMS
SynchronousResponseQueue		*CONNECTORNAME*/SYNCHRONOUSRESPONSEQUEUE	コンポーネント再始動	Delivery Transport は JMS

表 16. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
WireFormat	CwXML、CwBO	CwXML	エージェント再始動	Repository Directory が <REMOTE> でない場合は CwXML。Repository Directory が <REMOTE> であれば CwBO
WsifSynchronousRequest Timeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	WAS のみ
XMLNameSpaceFormat	short、long	short	エージェント再始動	WebSphere MQ Message Brokers および WAS のみ

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

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

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

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

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

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

AgentTraceLevel

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

ApplicationName

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

BrokerType

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

CharacterEncoding

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

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

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

ConcurrentEventTriggeredFlows

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

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

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

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

- `Maximum number of concurrent events` プロパティーの値を増加して、コラボレーションが複数のスレッドを使用できるように構成します。
- 宛先アプリケーションのアプリケーション固有コンポーネントが複数の要求を並行して実行できることを確認します。つまり、このコンポーネントがマルチスレッド化されているか、またはコネクター・エージェント並列処理を使用でき、複

数プロセスに対応するよう構成されている必要があります。Parallel Process Degree 構成プロパティーに、1 より大きい値を設定します。

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

ContainerManagedEvents

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

デフォルト値は No value です。

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

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

また、MimeType、DHClass、および DataHandlerConfigMOName (オプショナル) プロパティーを設定したデータ・ハンドラーも構成する必要があります。これらのプロパティーの値を設定するには、Connector Configurator の**「データ・ハンドラー」**タブを使用します。「データ・ハンドラー」タブの値のフィールドは、ContainerManagedEvents を JMS に設定した場合にのみ表示されます。

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

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

ControllerStoreAndForwardMode

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

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

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

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

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

デフォルト値は true です。

ControllerTraceLevel

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

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

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

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

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

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

WebSphere MQ および IDL

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

- 非同期 (ASYNC) 通信:
  WebSphere MQ を使用すると、アプリケーション固有のコンポーネントは、サーバーが利用不能である場合でも、イベントをポーリングして永続的に格納することができます。
- サーバー・サイド・パフォーマンス:
  WebSphere MQ を使用すると、サーバー・サイドのパフォーマンスが向上します。最適化モードでは、WebSphere MQ はイベントへのポインターのみをリポジトリー・データベースに格納するので、実際のイベントは WebSphere MQ キュー内に残ります。これにより、サイズが大きい可能性のあるイベントをリポジトリー・データベースに書き込む必要がありません。

- エージェント・サイド・パフォーマンス:
  WebSphere MQ を使用すると、アプリケーション固有のコンポーネント側のパフォーマンスが向上します。WebSphere MQ を使用すると、コネクターのポーリング・スレッドは、イベントを選出した後、コネクターのキューにそのイベントを入れ、次のイベントを選出します。この方法は IDL よりも高速で、IDL の場合、コネクターのポーリング・スレッドは、イベントを選出した後、ネットワーク経由でサーバー・プロセスにアクセスしてそのイベントをリポジトリー・データベースに永続的に格納してから、次のイベントを選出する必要があります。

JMS

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

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

このプロパティーは、false に設定することもできます。

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

FaultQueue

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

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

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

デフォルト値は 128M です。

JvmMaxNativeStackSize

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

デフォルト値は 128K です。

JvmMinHeapSize

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

デフォルト値は 1M です。

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

デフォルト値は crossworlds.queue.manager です。

jms.NumConcurrentRequests

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

デフォルト値は 10 です。

jms.Password

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

デフォルトはありません。

jms.UserName

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

デフォルトはありません。

ListenerConcurrency

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

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

Locale

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

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、以下のように説明されます。

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

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

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

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

LogAtInterchangeEnd

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

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

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

MaxEventCapacity

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

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

MessageFileName

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

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

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

MonitorQueue

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

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

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

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

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

デフォルト値は false です。

OADMaxNumRetry

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

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

デフォルト値は 1000 です。

OADRetryTimeInterval

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

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

デフォルト値は 10 です。

PollEndTime

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

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

PollFrequency

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

- ポーリング・アクション間のミリ秒数。

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

デフォルト値は 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。このプロパティーが使用されるかどうかを特定のコネクターについて判別するには、該当するアダプター・ガイドのインストールと構成についての章を参照してください。

PollQuantity

コネクタ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。

PollStartTime

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

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

RequestQueue

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

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

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

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

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

ResponseQueue

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

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

RestartRetryCount

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

デフォルト値は 3 です。

RestartRetryInterval

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

デフォルト値は 1 です。

SourceQueue

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

JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、61 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

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

WsifSynchronousRequest Timeout

WAS 統合ブローカーでのみ使用されます。

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

デフォルト値は 0 です。

XMLNameSpaceFormat

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

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

デフォルト値は short です。

付録 B. Connector Configurator

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

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

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

注:

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

この付録では、次のトピックについて説明します。

Connector Configurator の概要

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

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

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

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

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

Connector Configurator の実行モードと使用する構成ファイルのタイプは、実行する統合ブローカーによって異なる場合があります。例えば、使用している統合ブローカーが WMQI の場合、Connector Configurator を System Manager から実行するのではなく、直接実行します (72 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、73 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

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

Connector Configurator の始動

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

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

スタンドアロン・モードでの Configurator の実行

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

これを行うには、以下のステップを実行します。

- **「スタート」>「プログラム」**から、**「IBM WebSphere InterChange Server」>「IBM WebSphere Business Integration Toolset」>「開発」>「Connector Configurator」**をクリックします。
- **「ファイル」>「新規」>「構成ファイル」**を選択します。

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

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (78 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

1. 「System Manager」ウィンドウの「コネクター」フォルダーで構成ファイルを選択し、右クリックします。Connector Configurator が開き、この構成ファイルの統合ブローカー・タイプおよびファイル名が上部に表示されます。
2. 「標準のプロパティー」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、または既存のファイルをテンプレートとして使用します。

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

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。

テンプレートは以下のように作成します。

1. **「ファイル」>「新規」>「コネクタ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. 以下のフィールドを含む**「コネクター固有プロパティー・テンプレート (Connector-Specific Property Template)」**ダイアログ・ボックスが表示されます。
   - **「テンプレート」、「名前」**

     このテンプレートが使用されるコネクター (またはコネクターのタイプ) を表す固有の名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
   - **「旧テンプレート」、「変更する既存のテンプレートを選択してください」**

     「テンプレート名」表示に、現在使用可能なすべてのテンプレートの名前が表示されます。
   - テンプレートに含まれているコネクタ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。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。
3. **「テンプレート名」**表示からテンプレートを選択し、その名前を**「名前の検索 (Find Name)」**フィールドに入力し (または**「テンプレート名」**で自分の選択項目を強調表示し)、**「次へ」**をクリックします。

ご使用のコネクタ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。

一般特性の指定

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

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

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

値の指定

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

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

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

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

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

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

依存関係の設定

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

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

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

   == (等しい)

   != (等しくない)

   > (より大)

   < (より小)

>= (より大か等しい)

<= (より小か等しい)

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

新しい構成ファイルを作成

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

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。

- Connector Configurator のホーム・メニューで、**「ファイル」>「新規」>「コネクター構成」**をクリックします。**「新規コネクター」**ダイアログ・ボックスが表示されます。
- **「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入力します。

また、以下の作業も実行できます。

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

コネクター固有のテンプレートからの構成ファイルの作成

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

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

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

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

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

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

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

  使用するテンプレートを選択し、**「OK」**をクリックします。

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

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

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

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

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

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

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

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

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

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

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

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

構成ファイルの完成

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

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

1. **「標準のプロパティー」**タブで、BrokerType プロパティーの値フィールドを選択します。ドロップダウン・メニューで、値 ICS、WMQI、または WAS を選択します。
2. 選択したブローカーに関連付けられているプロパティーが「標準のプロパティー」タブに表示されます。ここでファイルを保管するか、または 82 ページの『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。
3. 構成が完了したら、**「ファイル」>「保管」>「プロジェクトに」**を選択するか、または**「ファイル」>「保管」>「ファイルに」**を選択します。

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

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

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

構成ファイル・プロパティーの設定

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

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

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

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

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

- 関連マップ
- リソース
- メッセージング (該当する場合)

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

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

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

「標準のプロパティー」と**「コネクター固有プロパティー (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。
- **「更新メソッド」**フィールドは通知を行うもので、構成できません。このフィールドでは、値が変更されたプロパティーをアクティブにするために必要なアクションを指定します。

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下の手順を実行します。

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

ー) で入力した値は、次のカテゴリーに移動しても保持されます。ウィンドウを閉じるときに、すべてのカテゴリーで入力した値を一括して保管するかまたは破棄するかを確認するプロンプトが出されます。

- 修正した値を保管するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「はい」**をクリックします。「ファイル」メニューまたはツールバーから**「保管」>「ファイルに」**をクリックする方法もあります。

アプリケーション固有の構成プロパティーの設定

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

1. グリッドの左上端の部分で右マウス・ボタン・クリックします。ポップアップ・メニュー・バーが表示されます。プロパティーを追加するときは**「追加」**をクリックします。子プロパティーを追加するには、親の行番号で右マウス・ボタンをクリックし、**「子を追加」**をクリックします。
2. プロパティーまたは子プロパティーの値を入力します。
3. プロパティーを暗号化するには、**「暗号化」**ボックスを選択します。
4. 80 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。

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

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

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

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

更新メソッド

付録『コネクターの*標準構成プロパティー*』の 54 ページの『プロパティー値の設定と更新』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

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

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

ご使用のブローカーが ICS の場合

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

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

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

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

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

ご使用のブローカーが WAS の場合

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

独立型方式で作業している (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。

関係付けられたマップ (ICS のみ)

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

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

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

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

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的**

  場合によっては、関連マップを明示的にバインドすることが必要になります。

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

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

以下の手順を実行して、マップを明示的にバインドします。

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

リソース (ICS)

「リソース」タブでは、コネクター・エージェントがコネクター・エージェント並列処理を使用して、同時に複数のプロセスを処理するかどうか、またどの程度処理するかを決定する値を設定することができます。
すべてのコネクターでこの機能がサポートされるわけではありません。複数のプロセスを使用するよりも複数のスレッドを使用する方が通常は効率的であるため、Java でマルチスレッドとして設計されたコネクター・エージェントを実行している場合、この機能を使用することはお勧めできません。

メッセージング (ICS)

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

トレース/ログ・ファイル値の設定

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

ログとトレースの値を変更するには、以下の手順を実行します。

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

     注: STDOUT オプションは、Windows プラットフォームで実行しているコネクターの**「トレース/ログ・ファイル」**タブでのみ使用できます。
   - ファイルに:
     ログ・メッセージまたはトレース・メッセージを指定されたファイルに書き込みます。ファイルを指定するには、ディレクトリー・ボタン (省略符号) をク

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

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

データ・ハンドラー

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

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

構成ファイルの保管

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

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

- System Manager から、統合コンポーネント・ライブラリーに `*.con` 拡張子付きファイルとして保管します。
- System Manager から、指定したディレクトリーに `*.con` 拡張子付きファイルとして保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに `*.cfg` 拡張子付きファイルとして保管します。

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

構成ファイルの変更

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

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

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

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

構成の完了

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

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

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

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

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

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

「CharacterEncoding」および「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>
```

特記事項

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

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

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

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

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

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

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

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

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

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

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

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

プログラミング・インターフェース情報

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

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

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

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

商標

以下は、IBM Corporation の商標です。

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

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

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

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標または登録商標です。

WebSphere Business Integration Adapter Framework V2.4.0

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