

**IBM WebSphere Business Integration  
Adapters**



**Adapter for WebSphere MQ 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、105 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 バージョン 2.5.0、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原 典：	IBM WebSphere Business Integration Adapters Adapter for WebSphere MQ 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

© Copyright International Business Machines Corporation 2000, 2003. All rights reserved.

© Copyright IBM Japan 2004

# 目次

本書について	v
対象読者	v
本書の前提条件	v
関連文書	v
表記上の規則	vi
<b>本リリースの新機能</b>	<b>vii</b>
リリース 2.5.x での新機能	vii
リリース 2.4.x での新機能	vii
リリース 2.3.x での新機能	vii
リリース 2.2.x での新機能	vii
リリース 2.1.x での新機能	viii
リリース 1.5.x での新機能	viii
リリース 1.4.x での新機能	ix
リリース 1.3.x での新機能	ix
<b>第 1 章 概要</b>	<b>1</b>
Adapter for WebSphere MQ の環境	2
コネクタ・アーキテクチャー	4
アプリケーションとコネクタ間の通信方法	5
イベント処理	6
保証付きイベント・デリバリー	10
ビジネス・オブジェクト要求	11
動詞の処理	11
ロケール依存データの処理	16
共通の構成タスク	17
<b>第 2 章 アダプターのインストールおよび構成</b>	<b>21</b>
インストール作業の概要	21
アダプターおよび関連ファイルのインストール	21
インストール済みファイルの構造	21
コネクタ構成	23
複数コネクタ・インスタンスの作成	29
キューの Uniform Resource Identifier (URI)	30
メタオブジェクト属性の構成	32
始動ファイルの構成	49
始動	49
コネクタの停止	50
<b>第 3 章 ビジネス・オブジェクトの作成および変更</b>	<b>53</b>
アダプターのビジネス・オブジェクトの構造	53
エラー処理	56
トレース	57
<b>第 4 章 トラブルシューティング</b>	<b>59</b>
始動時の問題	59
イベント処理	59
<b>付録 A. コネクタの標準構成プロパティ</b>	<b>61</b>
新規プロパティと削除されたプロパティ	61

標準コネクタ・プロパティの構成 . . . . .	61
標準プロパティの要約 . . . . .	63
標準構成プロパティ . . . . .	66
<b>付録 B. Connector Configurator . . . . .</b>	<b>79</b>
Connector Configurator の概要 . . . . .	79
Connector Configurator の始動 . . . . .	80
System Manager からの Configurator の実行 . . . . .	81
コネクタ固有のプロパティ・テンプレートの作成 . . . . .	81
新しい構成ファイルを作成 . . . . .	84
既存ファイルの使用 . . . . .	85
構成ファイルの完成 . . . . .	86
構成ファイル・プロパティの設定 . . . . .	87
構成ファイルの保管 . . . . .	94
構成ファイルの変更 . . . . .	95
構成の完了 . . . . .	95
グローバル化環境における Connector Configurator の使用 . . . . .	95
<b>付録 C. チュートリアル . . . . .</b>	<b>97</b>
チュートリアルについて . . . . .	97
始める前に . . . . .	98
環境のセットアップ . . . . .	98
シナリオの実行 . . . . .	100
<b>特記事項 . . . . .</b>	<b>105</b>
プログラミング・インターフェース情報 . . . . .	106
商標 . . . . .	107

---

## 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。

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

---

## 対象読者

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

---

## 本書の前提条件

本書の読者は、WebSphere Business Integration システム、ビジネス・オブジェクトとコラボレーションの開発、および WebSphere MQ アプリケーションについて十分な知識と経験を持っている必要があります。

---

## 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、「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 インプリメンテーション・ガイド」への参照が多数含まれています。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。

- アダプターの一般情報が必要な場合、アダプターを WebSphere Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) とともに使用する場合は、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>

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

---

## 表記上の規則

本書では、以下の規則を使用します。

---

Courier フォント	コマンド名、ファイル名、ユーザーの入力した情報、システムが画面に出力した情報など、記述されたとおりの値を示します。
太字	初出語を示します。
イタリック、イタリック青のアウトライン	変数名または相互参照を示します。 青のアウトラインは、マニュアルをオンラインで表示するときのみ見られるもので、相互参照用のハイパーリンクを示します。アウトラインの内側をクリックすることにより、参照先オブジェクトにジャンプできます。
{ }	構文の記述行の場合、中括弧 {} で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
[ ]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプション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入力できることを示します。
< >	命名規則により、1 つの名前の各エレメントを個々に判別できるようにするために、不等号括弧で囲みます。例えば、<server_name><connector_name>tmp.log のように使用します。
/, ¥	本書では、ディレクトリー・パスに円記号 (¥) を使用します。UNIX システムの場合には、円記号 (¥) をスラッシュ (/) に置き換えてください。すべての製品のパス名は、ご使用のシステムで製品がインストールされているディレクトリーを基準とした相対パス名です。
%text% と \$text	% 記号で囲まれたテキストは、Windows の text システム変数またはユーザー変数の値を示します。UNIX 環境の場合、これに相当する表記は \$text になります。これは、text UNIX 環境変数の値を示します。
ProductDir	製品のインストール先ディレクトリーを表します。

---

---

## 本リリースの新機能

---

### リリース 2.5.x での新機能

コネクタは、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 での新機能

アダプターは、WebSphere Application Server を統合ブローカーとして使用できるようになりました。詳しくは、2 ページの『ブローカーとの互換性』を参照してください。

コネクタは、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 での新機能

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」は現在は「System Manager」となり、「CrossWorlds InterChange Server」は現在は「WebSphere InterChange Server」となっています。

データ・ハンドラーを入力キューと関連付けることができるようになりました。詳しくは、36 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。

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、10 ページの『保証付きイベント・デリバリー』を参照してください。

---

### リリース 2.2.x での新機能

InProgress キューは不要になりました。使用不可に設定できます。詳しくは、28 ページの『InProgressQueue』を参照してください。

コネクタは、WebSphere MQ 5.1、5.2、および 5.3 を介したアプリケーションとのインターオペラビリティをサポートします。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。

コネクタに、ビジネス・オブジェクト処理のための UseDefaults プロパティが追加されました。詳しくは、29 ページの『UseDefaults』を参照してください。

データ・ハンドラーが明示的にビジネス・オブジェクトに対して動詞を割り当てていない場合、コネクタが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、26 ページの『DefaultVerb』を参照してください。

ReplyToQueue は、ReplyToQueue コネクタ・プロパティではなく動的子メタオブジェクトを介して指定できるようになりました。詳しくは、43 ページの『JMS ヘッダー、WebSphere MQ メッセージ・プロパティ、および動的子メタオブジェクトの属性』を参照してください。

メッセージ選択子を使用して、識別やフィルター操作を行えます。あるいは、アダプターが特定の要求に対して応答メッセージを識別する方法を制御できます。この JMS 機能は同期要求処理にのみ適用されます。詳しくは、12 ページの『同期デリバリー』を参照してください。

---

## リリース 2.1.x での新機能

コネクタは国際化されました。詳細については、16 ページの『ロケール依存データの処理』と 61 ページの『付録 A. コネクタの標準構成プロパティ』を参照してください。

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。

**注:** 保証付きイベント・デリバリー機能を構成するには、ICS のリリース 4.1.1.2 をインストールする必要があります。

---

## リリース 1.5.x での新機能

IBM WebSphere Business Integration Adapter for WebSphere MQ には、WebSphere MQ 用のコネクタが含まれます。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あり、データ・ルーティングなどのサービスを提供します。アダプターには、以下の要素が含まれます。

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

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



**重要:** コネクタ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、コネクタと ICS バージョン 4.1.1 を併用しないでください。

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

---

## リリース 1.4.x での新機能

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

- 以前のバージョンの WebSphere MQ 対応コネクタでは、WebSphere MQ メッセージと CrossWorlds ビジネス・オブジェクト間のデータ変換に使用するデータ・ハンドラーが、コネクタ・プロパティ `DataHandlerConfigMO` および `DataHandlerMimeType` によって決定されました。このため、異なるデータ・フォーマットを処理するためには、コネクタのインスタンスが複数必要でした。リリース 1.4.x では、コネクタの静的メタオブジェクトまたはビジネス・オブジェクトの動的子メタオブジェクトの要求内で、これらのプロパティをオプションで指定できるようになりました。詳細については、32 ページの『メタオブジェクト属性の構成』を参照してください。
- `DataEncoding` メタオブジェクト属性 (静的または動的) が追加されたことにより、メッセージ・タイプ (テキストまたはバイナリー) とエンコード・タイプを指定できるようになりました。詳細については、32 ページの『メタオブジェクト属性の構成』を参照してください。

---

## リリース 1.3.x での新機能

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

- 外部アプリケーションからコネクタに対して発行された同期要求のサポート。詳細については、7 ページの『同期イベント処理』を参照してください。
- `CollaborationName` プロパティがメタオブジェクトに追加されました。32 ページの『メタオブジェクト属性の構成』を参照してください。
- `DoNotReportBusObj` プロパティがメタオブジェクトに追加されました。32 ページの『メタオブジェクト属性の構成』を参照してください。



---

## 第 1 章 概要

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

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

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

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

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

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

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

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

---

## Adapter for WebSphere MQ の環境

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

- 『ブローカーとの互換性』
- 3 ページの『アダプター・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』

### ブローカーとの互換性

アダプターが使用するアダプター・フレームワークは、アダプターと通信する統合ブローカーのバージョンとの互換性を備えている必要があります。Adapter for WebSphere MQ 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・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、<sup>1</sup>および 5.3 を介したアプリケーションとのインターオペラビリティをサポートします。したがって、次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・テキストを配送できるように、国際化されています。コネクターが、ある文字コード・セットを使用する場所から別の文字コード・セットを使用する場所にデータを転送する場合、文字変換を実行してデータの意味を保持します。

Java 仮想マシン (JVM) 内の Java ランタイム環境では、データは Unicode 文字コード・セットで表現されます。Unicode には、ほとんどの既知の文字コード・セット (1 バイト系とマルチバイト系をいずれも含む) に対応できるエンコード方式が組み込まれています。WebSphere Business Integration システムのコンポーネントの大部分は Java で作成されています。したがって、ほとんどのインテグレーション・コンポーネント間で、文字変換を行わずにデータを転送できます。

エラー・メッセージや通知メッセージを国や地域に応じた適切な言語で記録するには、該当する環境の `Locale` 標準構成プロパティを設定します。構成プロパティの詳細については、61 ページの『付録 A. コネクターの標準構成プロパティ』を参照してください。

---

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。

---

## コネクタ・アーキテクチャー

コネクタ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、保証付きイベント・デリバリーも可能になります。

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

コネクタはキューから 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。

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

メッセージの処理に使用されるビジネス・オブジェクトのタイプと動詞は、WebSphere MQ メッセージ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。コネクタは、メタオブジェクト・エントリ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築し、WebSphere MQ メッセージ・ヘッダーの FORMAT フィールドのテキストに関連付けます。

オプションで動的メタオブジェクトを構築し、コネクタに渡されるビジネス・オブジェクトの子として追加することもできます。この子メタオブジェクトの値は、コネクタ全体に対して指定されている静的メタオブジェクトの値をオーバーライドします。子メタオブジェクトが定義されていない場合、または子メタオブジェクトが必要な変換プロパティを定義していない場合、デフォルトでは、コネクタは静的メタオブジェクトの値を調べます。1 つの静的コネクタ・メタオブジェクトの代わりに、またはその補足として、1 つ以上の動的子メタオブジェクトを指定できます。

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

入力キューからメッセージが検索されると、コネクタは、メッセージ・ヘッダーに含まれる FORMAT フィールドに関連付けられているビジネス・オブジェクト名を調べます。次に、そのビジネス・オブジェクトの新しいインスタンスと共に、メッセージの本体がデータ・ハンドラーに渡されます。フォーマットに関連付けられているビジネス・オブジェクト名がない場合は、メッセージの本体だけがデータ・ハ

ンドラーに渡されます。ビジネス・オブジェクトにメッセージの内容が正常に取り込まれると、コネクターはそのビジネス・オブジェクトがサブスクライブされているかどうかをチェックしてから、`gotAppEvents()` メソッドを使用して `InterChange Server` にデリバリーします。

## アプリケーションとコネクター間の通信方法

コネクターは、IBM WebSphere MQ に実装されている Java Message Service (JMS) を使用して通信します。JMS は、エンタープライズ・メッセージング・システムにアクセスするためのオープン・スタンダード API です。JMS は、ビジネス・アプリケーションがビジネス・データとイベントを非同期的に送受信できるように設計されています。

### メッセージ要求

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

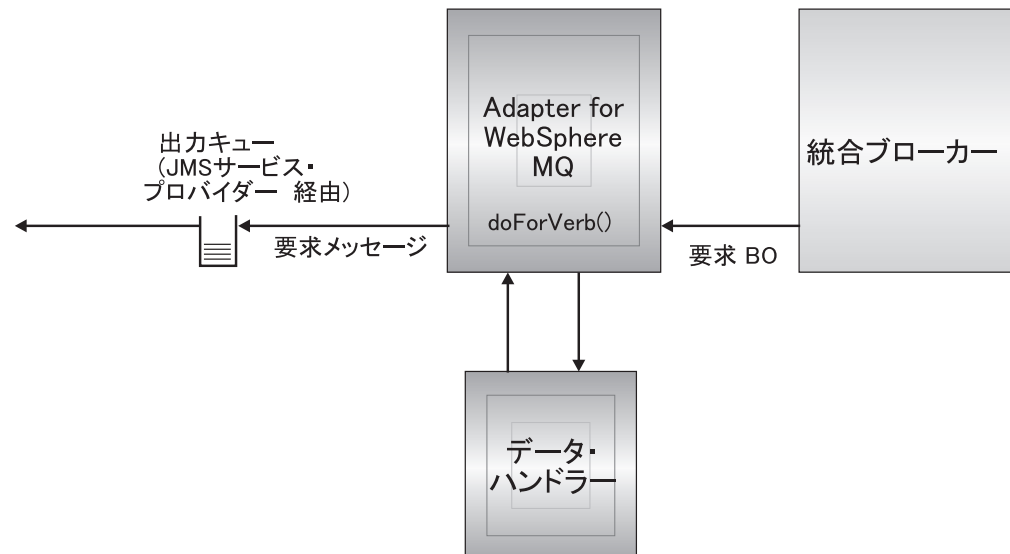


図 1. アプリケーションとコネクターの間の通信方法: メッセージ要求

### イベント・デリバリー

図 2 に、イベント・デリバリーの方向を示します。`pollForEvents()` メソッドは、次の該当するメッセージを入力キューから検索します。メッセージは実行中のキューに入れられ、処理が完了するまでキュー内に残ります。コネクターは最初に、静的メタオブジェクトまたは動的メタオブジェクトのいずれかを使用して、そのメッセージ・タイプがサポートされているかどうかを調べます。サポートされている場

合、コネクタは構成されているデータ・ハンドラーにメッセージを渡し、データ・ハンドラーがそれを WebSphere Business Integration システムのビジネス・オブジェクトに変換します。設定される動詞には、そのメッセージ・タイプに対して定義されている変換プロパティが反映されます。次に、コネクタは、そのビジネス・オブジェクトがコラボレーションによってサブスクライブされているかどうかを調べます。サブスクライブされている場合、getAppEvents() メソッドがビジネス・オブジェクトを InterChange Server にデリバリーし、実行中のキューからメッセージが削除されます。

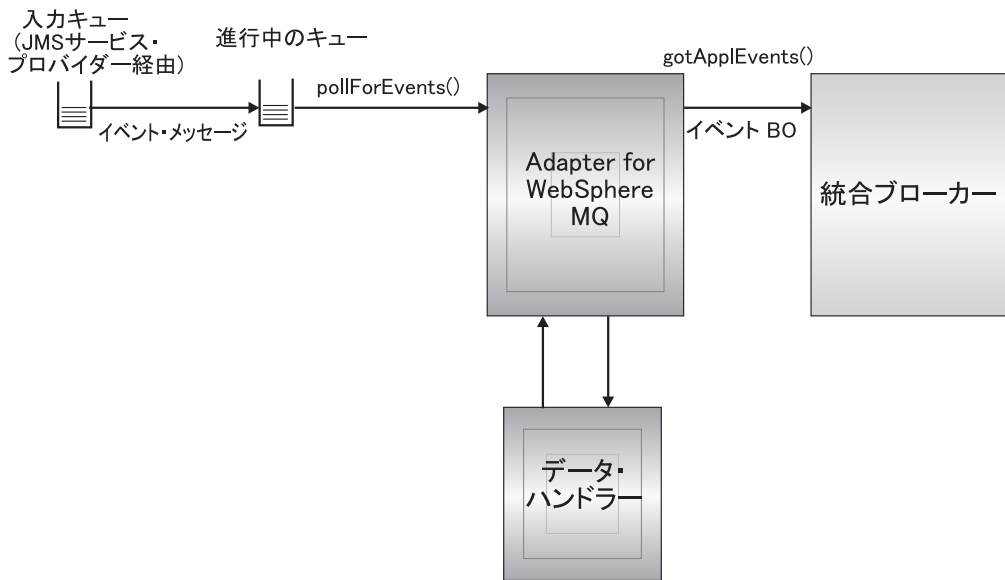


図2. アプリケーションとコネクタの間の通信方法: イベント・デリバリー

## イベント処理

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

## 検索

コネクタは、pollForEvents() メソッドを使用して MQ キューからメッセージを定期的にポーリングします。メッセージを検出すると、コネクタはそれを MQ キューから検索して調べ、メッセージのフォーマットを判別します。判別されたフォーマットがコネクタの静的オブジェクトで定義されている場合、コネクタは、メッセージの本体とそのフォーマットに関連付けられているビジネス・オブジェクトの新しいインスタンスの両方を、構成されているデータ・ハンドラーに渡します。データ・ハンドラーはビジネス・オブジェクトを取り込み、動詞を指定すると想定されています。判別されたフォーマットが静的メタオブジェクトで定義されていない場合、コネクタはメッセージの本体のみをデータ・ハンドラーに渡します。データ・ハンドラーはメッセージに対する正しいビジネス・オブジェクトを判



別して作成し、取り込むと想定されています。イベント失敗のシナリオについては、56 ページの『エラー処理』を参照してください。

コネクタは、最初に入力キューとのトランザクション・セッションを開いて、メッセージを処理します。このトランザクション・アプローチを使用すると、コネクタがビジネス・オブジェクトを正常にサブミットしたにもかかわらず、キューでトランザクションをコミットできなかった場合に、コラボレーションにビジネス・オブジェクトが 2 回デリバリーされてしまう可能性が若干あります。この問題を回避するために、コネクタはすべてのメッセージを実行中のキューに移動します。その結果、メッセージは、処理が完了するまでキュー内に保留されます。処理中にコネクタが予期しないエラーでシャットダウンした場合、メッセージは元の入力キューには戻されず、実行中のキュー内に残されます。

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

## 同期イベント処理

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

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

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

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

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

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

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

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

次の表に、コネクタから要求発行者に返送されるレポートの構造を示します。

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

## リカバリー

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

### Fail on startup

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

### Reprocess

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

## Ignore

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

## Log error

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

## アーカイブ

コネクターのプロパティ `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、53 ページの『第 3 章 ビジネス・オブジェクトの作成および変更』の 56 ページの『エラー処理』を参照してください。

**注:** JMS 規則により、検索したメッセージを即時に別のキューに送信することはできません。メッセージをアーカイブして再デリバリーできるようにするために、コネクターは、オリジナルのメッセージから本体とヘッダー（該当する場合のみ）を複製した第 2 のメッセージを最初に生成します。JMS サービス・プロバイダーとの競合を避けるため、JMS に必須のフィールドのみが複製されます。したがって、`format` フィールドは、アーカイブまたは再デリバリーされるメッセージにコピーされる唯一の追加メッセージ・プロパティとなります。

---

## 保証付きイベント・デリバリー

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

**注:** 保証付きイベント・デリバリー機能を使用しないと、コネクターがイベントをパブリッシュして（コネクターが `pollForEvents()` メソッド内の `gotAppEvent()` メソッドを呼び出して）から、イベント・レコードを削除してイベント・ストアを更新する（または「イベント通知済み」状況に更新する）までの間に、障害の可能性を示す短い間が空きます。このすき間で障害が発生すると、イベントは送信されますが、イベント・レコードはイベント・ストアで「進行中」状況の

ままになっています。コネクタは再始動時に、このイベント・ストアに残されたイベント・レコードを検出して送信するので、イベントが 2 回送信されることとなります。

保証付きイベント・デリバリー機能を、JMS イベント・ストアあり、またはなしで、JMS 対応コネクタのために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を構成するには、「コネクタ開発ガイド (Java 用)」の説明を参照してください。

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

---

## ビジネス・オブジェクト要求

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。コネクタは、構成されているデータ・ハンドラーを使用してビジネス・オブジェクトを WebSphere MQ メッセージに変換し、発行します。データ・ハンドラーについての要件を除いては、処理されるビジネス・オブジェクトのタイプに関する要件はありません。

---

## 動詞の処理

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

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

**注:** Create 動詞、Update 動詞、および Delete 動詞を持つビジネス・オブジェクトは、非同期的にも同期的にも送信できます。デフォルト・モードは非同期送信です。コネクタは、Retrieve 動詞、Exists 動詞、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。

## 非同期デリバリー

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

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

## 同期デリバリー

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

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

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

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

Report フィールドは、受信側アプリケーションから肯定処理レポートと否定処理レポートの両方の返送が予測されることを示すために設定されます。メッセージを発行したスレッドは、受信側アプリケーションが要求を処理できたかどうかを示す応答メッセージを待ちます。

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

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

\* は、IBM によって定義される定数を示します。

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

WebSphere MQ フィードバック・コード	対応する CrossWorlds 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
MQFB_APPL_FIRST + 8	APP_RESPONSE_TIMEOUT (この応答後、コネクタ・エージェントは即時に終了します)

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

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

コネクタは、応答メッセージを検索すると、デフォルトでは、応答の correlationID を要求メッセージの messageID と突き合わせます。続いて、要求を発行したスレッドに通知を送信します。コネクタは、応答の feedback フィールド

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

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

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

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

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

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

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

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

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

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

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

これで、応答メッセージが識別されます。



また、次のように、複数の置換を指定することも可能です。

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

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

次のいずれかの状況が発生すると、実行時にエラーが報告されます。

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

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

コネクターはこのメッセージ選択子を次のように解決します。

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

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

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

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

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

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

**カスタム・フィードバック・コードの作成:** コネクター・プロパティー FeedbackCodeMappingMO を指定することにより、WebSphere MQ フィードバック・コードを拡張してデフォルトの解釈をオーバーライドすることができます。このプロパティーを使用すると、WebSphere Business Integration システム固有のすべての戻り状況値を WebSphere MQ フィードバック・コードにマップしたメタオブジ

エクトを作成できます。(メタオブジェクトを使用して) フィードバック・コードに割り当てられた戻り状況値は、InterChange Server に渡されます。詳細については、26 ページの『FeedbackCodeMappingMO』を参照してください。

## Retrieve、Exists、および Retrieve By Content

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

次の表に、これらの動詞に対応する応答メッセージを示します。

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

## ロケール依存データの処理

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

Java 仮想マシン (JVM) 内の Java ランタイム環境では、データは Unicode 文字コード・セットで表現されます。Unicode には、ほとんどの既知の文字コード・セット (1 バイト系とマルチバイト系をいずれも含む) に対応できるエンコード方式が組み込まれています。WebSphere Business Integration システムのコンポーネントの大部分は Java で作成されています。したがって、ほとんどのインテグレーション・コンポーネント間で、文字変換を行わずにデータを転送できます。

エラー・メッセージや通知メッセージを国や地域に応じた適切な言語で記録するには、該当する環境の Locale 標準構成プロパティを設定します。構成プロパティの詳細については、61 ページの『付録 A. コネクタの標準構成プロパティ』を参照してください。

---

## 共通の構成タスク

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

### アダプターのインストール

何をどこにインストールするかについての詳細は、21 ページの『第 2 章 アダプターのインストールおよび構成』を参照してください。

### コネクタ・プロパティの構成

コネクタには、標準構成プロパティとコネクタ固有の構成プロパティの 2 種類の構成プロパティがあります。一部のプロパティはデフォルト値を持っており、変更を加えなくても使用できます。また、一部のプロパティについては、コネクタを実行する前に値を設定する必要があります。詳細については、21 ページの『第 2 章 アダプターのインストールおよび構成』を参照してください。

Adapter for WebSphere MQ のコネクタ・プロパティを構成する際には、次のことを確認してください。

- コネクタ・プロパティ HostName に指定した値が、使用している WebSphere MQ サーバーのホストの対応する値に一致している。
- コネクタ・プロパティ Port に指定した値が、使用しているキュー・マネージャーのリスナーのポートの対応する値に一致している。
- コネクタ・プロパティ Channel に指定した値が、使用しているキュー・マネージャーのサーバー接続チャンネルと一致している。
- コネクタ・プロパティ InputQueue、InProgressQueue、ArchiveQueue、ErrorQueue、および UnsubscribeQueue のキュー URI が有効であり、実際に存在する。

### 通知なしで要求を送信するためのコネクタの構成

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

- 送信する要求を表しており、コネクタ向けに構成したデータ・ハンドラーとの互換性もあるビジネス・オブジェクトを作成します。
- 静的メタオブジェクトまたは動的メタオブジェクトを使用して、宛先のキューとフォーマットを指定します。静的メタオブジェクトと動的メタオブジェクトの詳細については、32 ページの『静的メタオブジェクト』と 39 ページの『動的子メタオブジェクト』を参照してください。
- (静的または動的) メタオブジェクト内のプロパティ ResponseTimeout を -1 に設定します。この設定では、コネクタは発行したビジネス・オブジェクトの戻りをチェックしません。
- 詳細については、11 ページの『Create、Update、および Delete』、32 ページの『メタオブジェクト属性の構成』、および 53 ページの『第 3 章 ビジネス・オブジェクトの作成および変更』を参照してください。

## 要求を送信して通知を取得するためのコネクターの構成

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

- コネクターが応答を待機する時間を指示するために正の `ResponseTimeout` 値を指定する点を除いて、17 ページの『通知なしで要求を送信するためのコネクターの構成』の説明にある手順に従います。
- コネクターが予期する応答メッセージの具体的な詳細については、11 ページの『Create、Update、および Delete』を参照してください。示されている要件を応答メッセージが満たしていない場合、コネクターはエラーを報告したり、応答メッセージを認識できなかったりする可能性があります。32 ページの『メタオブジェクト属性の構成』と 53 ページの『第 3 章 ビジネス・オブジェクトの作成および変更』も参照してください。

## 静的メタオブジェクトの構成

静的メタオブジェクトは、ユーザーがビジネス・オブジェクトに関して指定したアプリケーション固有の情報と、コネクターによるビジネス・オブジェクトの処理方法についての情報を格納します。静的メタオブジェクトは、コネクターに、ビジネス・オブジェクトを処理するために必要なすべての情報を、コネクターの始動時に提供します。

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

- 32 ページの『静的メタオブジェクト』の手順に従います。
- コネクター固有のプロパティ `ConfigurationMetaObject` 内で静的メタオブジェクトの名前を指定することにより、コネクターが静的メタオブジェクトにサブスクライブするようにします。詳細については、24 ページの『コネクター固有のプロパティ』を参照してください。

## 動的メタオブジェクトの構成

コネクターがシナリオに応じて異なるビジネス・オブジェクト処理を実行する必要がある場合は、動的メタオブジェクトを使用します。これは、ビジネス・オブジェクトに追加する子オブジェクトです。動的メタオブジェクトは、要求の処理方法をコネクターに (実行時に) 指示します。静的メタオブジェクトは、コネクターがビジネス・オブジェクトを処理するために必要なすべての情報を、コネクターに提供します。これに対して、動的メタオブジェクトは、特定のシナリオの処理を実行するために必要なロジックの追加部分だけを提供します。動的メタオブジェクトを作成および構成するには、次の作業を実行します。

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

コネクタは動的メタオブジェクトをチェックし、その情報を使用してビジネス・オブジェクトの処理方法を判別します。詳細については、39 ページの『動的子メタオブジェクト』を参照してください。

## MQMD フォーマットの構成

MQMD はメッセージ記述子です。MQMD には、メッセージがアプリケーション間で送信されるときにアプリケーション・データに添付される制御情報が格納されます。静的メタオブジェクトまたは動的メタオブジェクト内で、MQMD 属性 `OutputFormat` の値を指定する必要があります。詳細については、11 ページの『Create、Update、および Delete』を参照してください。

## キュー URI の構成

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

- すべてのキューを URI (Uniform Resource Identifier) として指定します。構文は次のとおりです。  
`queue://<キュー・マネージャー名>/<実際のキュー>`
- コネクタ固有の構成プロパティに、キュー・マネージャーのホストを指定します。
- ターゲット・アプリケーションが MQMD ヘッダーのみを予期していて、JMS クライアントが使用する拡張 MQRFH2 ヘッダーを処理できない場合は、`?targetClient=1` をキュー URI に付加します。詳細については、30 ページの『キューの Uniform Resource Identifier (URI)』と WebSphere MQ のプログラミング・ガイドを参照してください。

## データ・ハンドラーの構成

データ・ハンドラーを構成する方法は 2 つあります。

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

## 始動スクリプトの変更

コネクタの始動方法の詳細については、21 ページの『第 2 章 アダプターのインストールおよび構成』を参照してください。始動する前に、コネクタ・プロパティを構成する必要があります。また、始動ファイルも変更する必要があります。

- クライアント・ライブラリーの格納場所が指定されるように、`start_connector` スクリプトを変更してください。複数のバージョン、または現在使用している WebSphere MQ サーバーに対応しないバージョンのクライアント・ライブラリーをインストールしないようにしてください。詳細については、49 ページの『始動ファイルの構成』を参照してください。



---

## 第 2 章 アダプターのインストールおよび構成

- 『インストール作業の概要』
- 『アダプターおよび関連ファイルのインストール』
- 『インストール済みファイルの構造』
- 23 ページの『コネクタ構成』
- 30 ページの『キューの Uniform Resource Identifier (URI)』
- 32 ページの『メタオブジェクト属性の構成』
- 49 ページの『始動ファイルの構成』
- 49 ページの『始動』

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

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

- **統合ブローカーのインストール** この作業では、WebSphere Business Integration システムのインストールと統合ブローカーの始動を行います。作業の詳細については、使用するブローカーおよびオペレーティング・システムのインストール文書に説明があります。
- **アダプターおよび関連ファイルのインストール** この作業では、アダプターのファイルをソフトウェア・パッケージから使用システムにインストールします。『アダプターおよび関連ファイルのインストール』を参照してください。

---

### アダプターおよび関連ファイルのインストール

WebSphere Business Integration adapter 製品のインストールの詳細については、以下の WebSphere Business Integration Adapters Infocenter のサイトにある「*WebSphere Business Integration Adapters* インストール・ガイド」を参照してください。

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

---

### インストール済みファイルの構造

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

**注:** Windows 環境でも UNIX 環境でも、一般に、WebSphere MQ と JMS は異なるディレクトリーにインストールされています。例えば AIX システムの場合、デフォルトでは、WebSphere MQ は /var/mqm/ にインストールされ、JMS は /usr/mqm/java/lib にインストールされています。JMS のインストールを /var/mqm/java/lib にリダイレクトすることにより、ルーチン /usr に関連するシステム管理タスクによる削除を防止することができます。同様に Windows

でも、通常、WebSphere MQ は %Program Files%WebSphere MQ の下にインストールされ、JMS は %Program Files%IBM%MQSeries%Java の下にインストールされます。これに従って、WebSphere MQ コネクタースタートスクリプト内のクラスパスを変更してください。

## Windows のファイル構造

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

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

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

<i>ProductDir</i> のサブディレクトリー	説明
connectors%WebSphereMQ%CWWebSphereMQ.jar	WebSphere MQ コネクタによってのみ使用されるクラスを含む
connectors%WebSphereMQConnector%start_WebSphereMQ.bat	コネクタ (NT/2000) の始動スクリプト
connectors%messages%WebSphereMQConnector.txt	コネクタのメッセージ・ファイル
repository%WebSphereMQ%CN_WebSphereMQ.txt	コネクタのリポジトリー定義
connectors%WebSphereMQ%samples%LegacyContact%WebSphereMQConnector.cfg	WebSphere MQ 構成ファイルのサンプル
connectors%WebSphereMQ%samples%LegacyContact%PortConnector.cfg	ポート・コネクタ構成ファイルのサンプル
connectors%WebSphereMQ%samples%LegacyContact%Sample_WebSphereMQ_LegacyContact.xsd	スキーマのサンプル
connectors%WebSphereMQ%samples%LegacyContact%Sample_WebSphereMQ_MO_Config.xsd	メタオブジェクトのサンプル
connectors%WebSphereMQ%samples%LegacyContact%Sample_WebSphereMQ_MO_DataHandler.xsd	データ・ハンドラー・メタオブジェクトのサンプル
connectors%WebSphereMQ%samples%LegacyContact%Sample_WebSphereMQ_MO_DataHandler_DelimitedConfig.xsd	区切りデータ・ハンドラー・メタオブジェクトのサンプル
connectors%WebSphereMQ%samples%LegacyContact%Sample_WebSphereMQ_DynMO_Config.xsd	動的なメタオブジェクトのサンプル
connectors%WebSphereMQ%samples%LegacyContact%JMSPropertyPairs.xsd	JMS プロパティーのサンプル

**注:** すべての製品のパス名は、使用システムで製品がインストールされたディレクトリーを基準とした相対パス名です。

## UNIX のファイル構造

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

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

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

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



---

**ProductDir** のサブディレクトリー

connectors/WebSphereMQ/start\_WebSphereMQ.sh

connectors/messages/WebSphereMQ/Connector.txt  
repository/WebSphereMQ/CN\_WebSphereMQ.txt  
connectors/WebSphereMQ/samples/LegacyContact/WebSphereMQConnector.cfg  
connectors/WebSphereMQ/samples/LegacyContact/PortConnector.cfg  
connectors/WebSphereMQ/samples/LegacyContact/Sample\_WebSphereMQ\_LegacyContact.xsd  
connectors/WebSphereMQ/samples/LegacyContact/Sample\_WebSphereMQ\_MO\_Config.xsd  
connectors/WebSphereMQ/samples/LegacyContact/Sample\_WebSphereMQ\_MO\_DataHandler.xsd

connectors/WebSphereMQ/samples/LegacyContact/Sample\_WebSphereMQ\_MO\_DataHandler\_DelimitedConfig.xsd  
connectors/WebSphereMQ/samples/LegacyContact/Sample\_WebSphereMQ\_DynMO\_Config.xsd  
connectors/WebSphereMQ/samples/LegacyContact/JMSPropertyPairs.xsd

**説明**

コネクターのシステム始動スクリプト。このスクリプトは、汎用のコネクター・マネージャー・スクリプトから呼び出されます。

System Manager の「コネクター構成」画面をクリックすると、このコネクター・マネージャー・スクリプトのカスタマイズ済みラッパーがインストーラーによって作成されます。コネクターの始動および停止には、このカスタマイズされたラッパーを使用してください。

コネクターのメッセージ・ファイル  
コネクターのリポジトリ定義

WebSphere MQ 構成ファイルのサンプル  
ポート・コネクター構成ファイルのサンプル  
スキーマのサンプル  
メタオブジェクトのサンプル  
データ・ハンドラー・メタオブジェクトのサンプル  
区切りデータ・ハンドラー・メタオブジェクトのサンプル  
動的なメタオブジェクトのサンプル  
JMS プロパティのサンプル

---

**注:** すべての製品のパス名は、使用システムで製品がインストールされたディレクトリーを基準とした相対パス名です。

---

## コネクター構成

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

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

- Connector Configurator の説明と段階的な手順については、79 ページの『付録 B. Connector Configurator』を参照してください。
- 標準コネクター・プロパティーの説明については、24 ページの『標準コネクター・プロパティー』、および 61 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。
- コネクター固有のプロパティーの詳細については、24 ページの『コネクター固有のプロパティー』を参照してください。

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

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

## 標準コネクタ・プロパティ

標準構成プロパティにより、すべてのコネクタによって使用される情報が提供されます。標準構成プロパティの資料については、61 ページの『付録 A. コネクタの標準構成プロパティ』を参照してください。

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

## コネクタ固有のプロパティ

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

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

名前	指定可能な値	デフォルト値	必須
ApplicationPassword	ログイン・パスワード		いいえ
ApplicationUserName	ログイン・ユーザー ID		いいえ
ArchiveQueue	正常に処理されたメッセージのキューが送信されるキュー	queue://crossworlds. queuemanager/MQCONN.ARCHIVE	いいえ
CCSID	キュー・マネージャーの接続に使用する文字セット	null	いいえ
Channel	MQ サーバー・コネクタ・チャネル		はい
ConfigurationMetaObject	構成メタオブジェクトの名前		はい
DataHandlerClassName	データ・ハンドラー・クラス名	com.crossworlds.DataHandlers. text.xml	いいえ
DataHandlerConfigMO	データ・ハンドラー・メタオブジェクト	MO_DataHandler_Default	はい
DataHandlerMimeType	MIME ファイル・タイプ	text/xml	いいえ
DefaultVerb	コネクタがサポートする任意の動詞	Create	
ErrorQueue	未処理のメッセージ・キュー	queue://crossworlds. queuemanager/MQCONN.ERROR	いいえ
FeedbackCodeMappingMO	フィードバック・コード・メタオブジェクト		いいえ
HostName	WebSphere MQ サーバー		はい
InDoubtEvents	FailOnStartup Reprocess IgnoreLogError	Reprocess	いいえ
InputQueue	ポーリング・キュー	queue://crossworlds. queuemanager/MQCONN.IN	いいえ
InProgressQueue	進行中のイベント・キュー	queue://crossworlds. queuemanager/MQCONN.IN_PROGRESS	いいえ
PollQuantity	InputQueue プロパティ内で指定された各キューから検索されるメッセージの数	1	いいえ

名前	指定可能な値	デフォルト値	必須
Port	WebSphere MQ リスナー用に設定されたポート		はい
ReplyToQueue	コネクターからの要求発行時に応答メッセージが配信されるキュー	queue://crossworlds. queuemanager/MQCONN.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	queue://crossworlds. queuemanager/MQCONN.UNSUBSCRIBE	いいえ
UseDefaults	true または false	false	

## ApplicationPassword

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

デフォルト = 設定値なし

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

注 \*

## ApplicationUserName

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

デフォルト = 設定値なし

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

注 \*

## ArchiveQueue

正常に処理されたメッセージのコピーが送信されるキューです。

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

## CCSID

キュー・マネージャーの接続に使用する文字セット。このプロパティーの値は、キュー URI 内の CCSID プロパティーの値と一致する必要があります。30 ページの『キューの Uniform Resource Identifier (URI)』を参照してください。

デフォルト値はヌルです。

## Channel

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

デフォルト = 設定値なし

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

## ConfigurationMetaObject

コネクタの構成情報を含む静的なメタオブジェクトの名前です。

デフォルト = 設定値なし

## DataHandlerClassName

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

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

## DataHandlerConfigMO

構成情報を提供するために、データ・ハンドラーに渡されるメタオブジェクト。

デフォルト = `MO_DataHandler_Default`

## DataHandlerMimeType

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

デフォルト = `text/xml`

## DefaultVerb

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

デフォルト = `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.queue.manager/MQCONN.ERROR`

## FeedbackCodeMappingMO

メッセージの受信を InterChange Server に同期的に確認するために使用されるデフォルトのフィードバック・コードをオーバーライドして再割り当てするプロパティです。このプロパティを使用すると、フィードバック・コードを表示するために各属性名を解釈するときのメタオブジェクトを指定できます。フィードバック・コードの対応する値は、InterChange Server に渡される戻り状況値です。デフォルトのフィードバック・コードのリストは、12 ページの『同期デリバリー』を参照してください。コネクタは、WebSphere MQ 固有のフィードバック・コードを表す以下の属性値を受け入れます。

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

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

- SUCCESS
- FAIL
- APP\_RESPONSE\_TIMEOUT
- MULTIPLE\_HITS
- UNABLE\_TO\_LOGIN
- VALCHANGE
- VALDUPES

以下の表に、サンプル・メタオブジェクトを示します。

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

デフォルト = 設定値なし

## HostName

WebSphere MQ をホスティングしているサーバーの名前です。

デフォルト = 設定値なし

## InDoubtEvents

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

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

デフォルト = Reprocess

## InputQueue

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

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

コネクタはラウンドロビン方式でキューをポーリングし、各キューから *pollQuantity* で指定された値を最大数とするメッセージを検索します。例えば、

`pollQuantity` が 2 であり、`MyQueueA` に 2 件のメッセージがあり、`MyQueueB` に 1 件のメッセージがあり、`MyQueueC` に 5 件のメッセージがある場合は、コネクタは以下のようにメッセージを取得します。

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

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

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

### InProgressQueue

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

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

### PollQuantity

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

デフォルト = 1

### Port

WebSphere MQ リスナー用に設定されたポートです。

デフォルト = 設定値なし

### ReplyToQueue

コネクタからの要求発行時に応答メッセージが配信されるキューです。子動的メタオブジェクトの属性を使用して応答を無視することもできます。このような属性

の詳細については、43 ページの『JMS ヘッダー、WebSphere MQ メッセージ・プロパティ、および動的子メタオブジェクトの属性』を参照してください。

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

### UnsubscribedQueue

アンサブスクライブされたメッセージが送信されるキューです。

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

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

### UseDefaults

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

---

## 複数コネクタ・インスタンスの作成

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

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

### 新規ディレクトリの作成

コネクタ・インスタンスごとにコネクタ・ディレクトリを作成する必要があります。このコネクタ・ディレクトリの名前は、

`ProductDir%connectors%connectorInstance`

である必要があります。ここで、connectorInstance によりコネクタ・インスタンスを一意に識別できます。

コネクタに、コネクタ固有のメタオブジェクトがある場合は、コネクタ・インスタンス用のメタオブジェクトを作成する必要があります。メタオブジェクトをファイルとして作成する場合、

`ProductDir%repository%connectorInstance`

というディレクトリを作成して、そこにファイルを保管します。

## ビジネス・オブジェクト定義の作成

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

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

```
ProductDir¥repository¥initialConnectorInstance
```

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

## コネクタ定義の作成

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

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

## 始動スクリプトの作成

始動スクリプトを作成するには、次の作業を実行します。

1. 初期コネクタの始動スクリプトをコピーし、そのファイルに次のコネクタ・ディレクトリ名が含まれる名前を付ける。  
`dirname`
2. この始動スクリプトを 29 ページの『新規ディレクトリの作成』で作成したコネクタ・ディレクトリに置く。
3. 始動スクリプトのショートカットを作成する (Windows のみ)。
4. 初期コネクタのショートカット・テキストをコピーして、コマンド行から新規コネクタ・インスタンスに合うように初期コネクタ名を変更する。

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

カスタム・コネクタ作成の詳細については、「コネクタ開発ガイド (C++ 用)」または「コネクタ開発ガイド (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/MQCONN.IN?targetClient=1&priority=5
```

以下の表に、キュー URI のプロパティー名を示します。

プロパティー名	説明	値
<code>expiry</code>	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)。
<code>priority</code>	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、コネクタ自身のデフォルト値を使用できるように指定します。
<code>persistence</code>	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、コネクタ自身のデフォルト値を使用できるように指定します。
<code>CCSID</code>	アウトバウンド・メッセージをエンコードする文字セット	整数: WebSphere MQ の資料にリストされている有効な値。この値は、CCSID コネクタ固有の構成プロパティーの値と一致する必要があります。25 ページの『CCSID』を参照してください。
<code>targetClient</code>	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)
<code>encoding</code>	数値フィールドの表示方法	基本的な WebSphere MQ 資料に記載されている整数値。

**注:** アダプターは、MQMessage 内のデータの文字セット (CCSID) またはエンコード属性を制御できません。データ変換はデータがメッセージ・バッファーから検索されるかメッセージ・バッファーにデリバリーされる時に行われるため、コネクターは JMS の IBM WebSphere MQ インプリメンテーションに依存してデータ変換を行います (IBM WebSphere MQ Java クライアント・ライブラリーの資料を参照してください)。したがって、これらの変換は、ネイティブ WebSphere MQ API がオプション `MQGMO_CONVERT` を使用して実行する変換と双方向で等しくなければなりません。コネクターは、変換プロセスにおける差異または失敗を制御できません。コネクターは、特別な変更を必要とせず、WebSphere MQ によってサポートされるすべての CCSID またはエンコードの

メッセージ・データを検索できます。特定の CCSID またはエンコードのメッセージを送信するには、出力キューが完全修飾 URI であり、CCSID および encoding の値を指定している必要があります。コネクタはこの情報を WebSphere MQ に渡し、WebSphere MQ は MQMessage のデリバリーのためにデータをエンコードするときに (JMS API を介して) この情報を使用します。多くの場合、CCSID およびエンコードのサポートの欠如は、IBM の Web サイトから最新バージョンの IBM WebSphere MQ Java クライアント・ライブラリーをダウンロードすることによって解決できます。それでも CCSID およびエンコードに関する問題が解消されない場合は、IBM ソフトウェア・サポートに連絡し、代替の Java Virtual Machine を使用してコネクタを実行することを検討してください。

---

## メタオブジェクト属性の構成

WebSphere MQ 対応コネクタは、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。

- 静的なコネクタ・メタオブジェクト
- 動的な子メタオブジェクト

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

## 静的メタオブジェクト

WebSphere MQ の静的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 を含む Customer オブジェクトの変換プロパティを定義するには、Customer\_Create という名前の属性を作成します。属性のアプリケーション固有のテキストで、実際の変換プロパティを指定します。

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

以下の表で、メタオブジェクトのプロパティを説明します。

---

プロパティ名	説明
CorrelationID	このプロパティは要求処理中のアダプター動作にのみ影響を与えます。このプロパティの処理方法は、動的メタオブジェクト内の CorrelationID プロパティと同じです。詳しくは、47 ページの『非同期要求の処理』を参照してください。

---

プロパティ名	説明
CollaborationName	<p>CollaborationName は、ビジネス・オブジェクトと動詞の組み合わせに対する属性のアプリケーション固有のテキスト内で指定される必要があります。例えば、ユーザーが動詞 Create 付きのビジネス・オブジェクト Customer の同期要求を処理しようとしている場合、静的メタデータ・オブジェクトは Customer_Create という名前の属性を含んでいる必要があります。</p> <p>Customer_Create 属性は、名前と値のペアを含むアプリケーション固有のテキストを含んでいる必要があります。例えば、CollaborationName=MyCustomerProcessingCollab です。構文の詳細については、35 ページの『アプリケーション固有の情報』のセクションを参照してください。</p>
DataEncoding	<p>この条件が満たされていない場合は、コネクタが Customer ビジネス・オブジェクトに関する要求を同期的に処理しようとするランタイム・エラーが発生します。</p> <p><b>注:</b> このプロパティは、同期要求にのみ利用可能です。</p> <p>DataEncoding は、メッセージの読み取りおよび書き込みに使用されるエンコードです。このプロパティが静的メタオブジェクトで指定されていない場合、コネクタは特定のエンコードを使用せずにメッセージを読み取るようにします。動的子メタオブジェクトで定義された DataEncoding は、静的メタオブジェクトで定義された値をオーバーライドします。デフォルト値は Text です。この属性の値のフォーマットは、messageType[:enc] です。例えば、Text:ISO8859_1、Text:UnicodeLittle、Text、または Binary のようになります。このプロパティは内部的に InputFormat プロパティに関連します。InputFormat ごとに 1 つの DataEncoding のみを指定します。</p>
DataHandlerConfigMO	<p>構成情報を提供するために、データ・ハンドラーに渡されるメタオブジェクト。静的なメタオブジェクトに指定された場合、この値は DataHandlerConfigMO コネクタ・プロパティに指定された値をオーバーライドします。さまざまなビジネス・オブジェクト・タイプを処理するために各種のデータ・ハンドラーが必要な場合は、この静的なメタオブジェクトのプロパティを使用します。動的な子メタオブジェクトに指定された場合、このプロパティはコネクタ・プロパティと静的なメタオブジェクトのプロパティをオーバーライドします。データ形式が実際のビジネス・データに依存する可能性がある場合は、要求処理には動的な子メタオブジェクトを使用します。指定されるビジネス・オブジェクトは、コネクタによってサポートされている必要があります。</p>

プロパティ名	説明
DataHandlerMimeType	<p>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。静的なメタオブジェクトに指定された場合、この値は DataHandlerMimeType コネクター・プロパティに指定された値をオーバーライドします。さまざまなビジネス・オブジェクト・タイプを処理するために各種のデータ・ハンドラーが必要な場合は、この静的なメタオブジェクトのプロパティを使用します。動的な子メタオブジェクトに指定された場合、このプロパティはコネクター・プロパティと静的なメタオブジェクトのプロパティをオーバーライドします。データ形式が実際のビジネス・データに依存する可能性がある場合は、要求処理には動的な子メタオブジェクトを使用します。DataHandlerConfigM0 に指定されたビジネス・オブジェクトは、このプロパティの値に対応する属性を含める必要があります。</p>
DoNotReportBusObj	<p>オプションとして、ユーザーは DoNotReportBusObj プロパティを含めることができます。このプロパティを true に設定することにより、発行されるすべての PAN レポート・メッセージのメッセージ本体がブランクになります。このオプションは、要求発行者がビジネス・オブジェクトが正常に処理されたことの確認は必要としますが、ビジネス・オブジェクトの変更に関する通知は必要としない場合にお勧めします。このプロパティは NAN レポートには影響しません。</p> <p>静的メタオブジェクトにこのプロパティがない場合、コネクターはこのプロパティをデフォルトの false に設定し、メッセージ・レポートにビジネス・オブジェクトを取り込みます。</p> <p><b>注:</b> このプロパティは、同期要求にのみ利用可能です。InputFormat は、特定のビジネス・オブジェクトと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、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。このプロパティは内部的に DataEncoding プロパティに関連します。InputFormat ごとに 1 つの DataEncoding のみを指定します。このプロパティを設定するときは、デフォルトの変換プロパティを使用しないでください。デフォルトの変換プロパティの値は、着信メッセージをビジネス・オブジェクトに一致させるために使用されます。</p>
InputFormat	<p>OutputFormat は、指定されたビジネス・オブジェクトから作成されたメッセージで設定されます。OutputFormat が指定されていない場合、使用可能であれば入力フォーマットが使用されます。動的な子メタオブジェクトに定義された OutputFormat は、静的なメタオブジェクトに定義された値をオーバーライドします。</p>
OutputFormat	

プロパティ名	説明
InputQueue	<p>コネクタが、新しいメッセージを検出するためにポーリングする入力キュー。</p> <p><b>注:</b> コネクタ固有のプロパティにある <code>InputQueue</code> プロパティは、アダプターがポーリングするキューを定義します。これは、アダプターがポーリングするキューを決定するのに使用する唯一のプロパティです。静的 MO では、<code>InputQueue</code> プロパティおよび <code>InputFormat</code> プロパティは、アダプターが指定されたメッセージを特定のビジネス・オブジェクトにマップする条件として使用できます。データ・ハンドラーの <code>InputQueues</code> へのマッピングについては、36 ページの『データ・ハンドラーの <code>InputQueues</code> へのマッピング』を参照してください。</p>
OutputQueue	<p><code>OutputQueue</code> 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配信される出力キューです。動的な子メタオブジェクトに定義された <code>OutputQueue</code> は、静的なメタオブジェクトに定義された値をオーバーライドします。</p>
ResponseTimeout	<p>応答を待機した状態で、タイムアウトになるまでの時間をミリ秒で表します。このプロパティが未定義のままか、またはゼロよりも小さい値に設定されている場合、コネクタは応答を待機せず、SUCCESS を即時に戻します。動的な子メタオブジェクトに定義された <code>ResponseTimeout</code> は、静的なメタオブジェクトに定義された値をオーバーライドします。</p>
TimeoutFatal	<p>このプロパティが定義されていて値が <code>True</code> の場合、<code>ResponseTimeout</code> で指定された時間内に応答がないと、コネクタは <code>APP_RESPONSE_TIMEOUT</code> 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、<code>InterChange Server</code> に <code>APP_RESPONSE_TIMEOUT</code> を即時に戻します。これにより、<code>InterChange Server</code> はコネクタを終了します。動的な子メタオブジェクトに定義された <code>TimeoutFatal</code> は、静的なメタオブジェクトに定義された値をオーバーライドします。</p>

さらに、`Default` という名前の予約済みプロパティもメタオブジェクト内で定義できます。このプロパティが存在する場合は、こ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のデフォルト値が指定されます。

以下に、サンプル・メタオブジェクトを示します。

プロパティ名	アプリケーション固有のテキスト
デフォルト	<pre>DataEncoding=Text:UnicodeLittle; OutputFormat=CUST_OUT; OutputQueue=QueueA;ResponseTimeout=10000; TimeoutFatal=False</pre>

## アプリケーション固有の情報

アプリケーション固有の情報は、名前と値のペアで構成され、それらはセミコロンで区切られています。以下に例を示します。

```
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で InputQueue プロパティを使用することにより、データ・ハンドラーと入力キューを関連付けることができます。この機能は、異なる書式や変換要件を持つ複数の取引先と取り引きする場合に役立ちます。それには、以下の作業を行う必要があります。

1. コネクタ固有のプロパティ (27 ページの『InputQueue』参照) を使用して 1 つ以上の入力キューを構成します。
2. それぞれの入力キューごとに、キュー・マネージャーおよび入力キュー名を指定し、またアプリケーション固有情報にデータ・ハンドラーのクラス名および MIME タイプを指定する。

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

```
[Attribute]
Name = Cust_Create
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = InputQueue=//queue.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

コネクタは通常、メッセージ検索時に入力フォーマットを特定のビジネス・オブジェクトと動詞の組み合わせと付き合わせます。次に、コネクタはそのビジネス・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要求するビジネス・オブジェクトと対応しているかどうかを確認できます。

ただし、2 つ以上のビジネス・オブジェクトに同一の入力フォーマットが定義されている場合は、コネクタはデータ・ハンドラーにデータを渡す前にそのデータが表すビジネス・オブジェクトを判別することはできません。このような場合、コネクタはメッセージ内容のみをデータ・ハンドラーに渡してから、生成されるビジネス・オブジェクトに基づいた変換プロパティを検索します。したがって、データ・ハンドラーはメッセージ内容のみに基づいてビジネス・オブジェクトを判別する必要があります。

生成されるビジネス・オブジェクトの動詞が設定されていない場合、コネクタはなんらかの動詞を含む同じビジネス・オブジェクトに定義されている変換プロパティを検索します。変換プロパティのセットが 1 つだけ検出された場合、コネクタは特定の動詞を割り当てます。複数の変換プロパティが検出された場合、コネクタは動詞を区別できないため、メッセージは失敗します。

## 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を構成します。属性 Default はメタオブジェクトで定義されます。コネクタは以下の属性を持つ変換プロパティを使用します。

```
OutputQueue=CustomerQueue1;ResponseTimeout=5000;TimeoutFatal=true
```

この属性は、その他すべての変換プロパティのデフォルト値として使用されます。したがって、ある属性によって別の指定をされたり動的な子メタオブジェクト値によってオーバーライドされる場合を除いて、コネクタはすべてのビジネス・オブジェクトをキュー CustomerQueue1 に発行し、その後応答メッセージを待機します。5000 ミリ秒内に応答が到着しない場合、コネクタ はすぐに終了します。

**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。この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、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、データ・ハンドラーがそれらを識別できるようになっている必要があります (36 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)。Request 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。デフォルトの応答時間は、コネクタがタイムアウトまでに最大 10000 ミリ秒待機するようにオーバーライドされているので注意してください (応答が受信されない場合も終了します)。

```
[ReposCopy]
Version = 3.1.0
Repositories = 1cHyILNuPTc=
[End]
[BusinessObjectDefinition]
Name = Sample_M0
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。

コネクタは、コネクタに渡されるトップレベルのビジネス・オブジェクトに子として追加された動的メタオブジェクトから、変換プロパティを認識して読み取ります。この動的な子メタオブジェクトの属性値は、コネクタの構成に使用される静的なメタオブジェクトに指定可能であった変換プロパティと重複します。

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

**注:** コネクタは、同期イベント送達中の、動的子メタオブジェクトを使用したコラボレーション名の指定をサポートしていません。

前のセクションの表と次の表に、ビジネス・オブジェクト (Customer\_Create) の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の情報は名前と値のペアで構成され、それぞれのペアはセミコロンで区切られています。

プロパティ名	値
DataEncoding	Text:UnicodeLittle
DataHandlerMimeType*	text/delimited
OutputFormat	CUST_OUT
OutputQueue	QueueA
ResponseTimeout	10000
TimeoutFatal	False

\* コネクタ構成プロパティまたは静的メタオブジェクトに DataHandlerConfigMO が指定されていると想定されています。

コネクタは、受信されたトップレベル・ビジネス・オブジェクトのアプリケーション固有の情報を調べて、タグ `cw_mo_conn` が子メタオブジェクトを指定しているかどうかを判断します。子メタオブジェクトが指定されている場合、動的な子メタオブジェクトの値が静的なメタオブジェクトに指定された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

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

次の表に、動的子メタオブジェクトのポーリングのための構成の例を示します。

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

上記の表に示すように、動的子メタオブジェクトでは、追加の属性 `InputQueue` を定義できます。この属性は、特定のメッセージの検索元キューの名前を含みます。このプロパティが子メタオブジェクトで定義されていない場合は、このプロパティは取り込まれません。

シナリオ例:

- コネクターは、キュー `MyInputQueue` からフォーマット `CUST_IN` でメッセージを取得します。
- コネクターはこのメッセージを `Customer` ビジネス・オブジェクトに変換し、アプリケーション固有のテキストを調べてメタオブジェクトが定義されているかどうかを判断します。
- メタオブジェクトが定義されている場合、コネクターはこのメタオブジェクトのインスタンスを作成し、定義に基づいて `InputQueue` および `InputFormat` 属性に値を取り込んで、ビジネス・オブジェクトを使用可能なコラボレーションにパブリッシュします。

## 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、WebSphere MQ メッセージ・プロパティ、および動的子メタオブジェクトの属性

動的メタオブジェクトに属性を追加すると、メッセージ・トランスポートの詳細情報を取得したりメッセージ・トランスポートを詳細に制御したりすることができます。このような属性を追加すると、JMS プロパティを変更し、(アダプター・プロパティで指定されたデフォルト ReplyToQueue を使用せずに) 要求ごとに ReplyToQueue を制御したり、メッセージの CorrelationID を再ターゲットしたりすることができます。このセクションでは、これらの属性、および同期モードと非同期モードの両方におけるイベント通知と要求処理に対する影響について説明します。

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

表 1. 動的メタオブジェクト・ヘッダー属性

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

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

以下のセクションでは、これらの属性の解釈および使用について説明します。

**注:** 上記の属性はいずれも必須ではありません。ビジネス・プロセスに関連する動的メタオブジェクトには任意の属性を追加できます。

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

1. 属性の名前はセマンティック値を持ちません。
2. 属性のタイプは、JMS プロパティ・タイプに無関係に必ず String でなければなりません。
3. 属性のアプリケーション固有情報は、属性をマップする JMS メッセージ・プロパティの名前と形式を定義する 2 つの名前と値の組を含まなければなりません。

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

表 2. JMS プロパティ属性のアプリケーション固有情報

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

以下の図に、動的メタオブジェクトの属性 JMSProperties および JMS メッセージ・ヘッダーの 4 つのプロパティ (ID、GID、RESPONSE、および RESPONSE\_PERSIST) の定義を示します。属性のアプリケーション固有情報はそれぞれの名前およびタイプを定義します。例えば、属性 ID はタイプ String の JMS プロパティ ID にマップされます。

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

図3. 動的メタオブジェクトの JMS プロパティ属性

**非同期イベント通知:** ヘッダー属性を持つ動的メタオブジェクトがイベント・ビジネス・オブジェクトに存在する場合は、コネクタは、(メタオブジェクトにトランスポート関連のデータを設定するほかに) 以下のステップを実行します。

1. メタオブジェクトの CorrelationId 属性に、メッセージの JMSCorrelationID ヘッダー・フィールドで指定された値を設定します。
2. メタオブジェクトの ReplyToQueue 属性に、メッセージの JMSReplyTo ヘッダー・フィールドで指定されたキューを設定します。このヘッダー・フィールドはメッセージの Java オブジェクトによって表されるため、この属性にはキューの名前 (多くの場合は URI) が設定されます。
3. メタオブジェクトの DeliveryMode 属性に、メッセージの JMSDeliveryMode ヘッダー・フィールドで指定された値を設定します。
4. メタオブジェクトの Priority 属性に、メッセージの JMSPriority ヘッダー・フィールドを設定します。
5. メタオブジェクトの Destination 属性に、メッセージの JMSDestination ヘッダー・フィールドの名前を設定します。Destination はオブジェクトによって表されるため、この属性には Destination オブジェクトの名前が設定されます。
6. メタオブジェクトの Expiration 属性に、メッセージの JMSExpiration ヘッダー・フィールドの値を設定します。
7. メタオブジェクトの MessageID 属性に、メッセージの JMSMessageID ヘッダー・フィールドの値を設定します。
8. メタオブジェクトの Redelivered 属性に、メッセージの JMSRedelivered ヘッダー・フィールドの値を設定します。
9. メタオブジェクトの TimeStamp 属性に、メッセージの JMSTimeStamp ヘッダー・フィールドの値を設定します。
10. メタオブジェクトの Type 属性に、メッセージの JMSType ヘッダー・フィールドの値を設定します。
11. メタオブジェクトの UserID 属性に、メッセージの JMSXUserID プロパティ・フィールドの値を設定します。
12. メタオブジェクトの AppID 属性に、メッセージの JMSXAppID プロパティ・フィールドの値を設定します。
13. メタオブジェクトの DeliveryCount 属性に、メッセージの JMSXDeliveryCount プロパティ・フィールドの値を設定します。

14. メタオブジェクトの GroupID 属性に、メッセージの JMSXGroupID プロパティ・フィールドの値を設定します。
15. メタオブジェクトの GroupSeq 属性に、メッセージの JMSXGroupSeq プロパティ・フィールドの値を設定します。
16. メタオブジェクトの JMSProperties 属性に定義されたオブジェクトを検証します。アダプターは、メッセージの対応するプロパティの値をこのオブジェクトの各属性に設定します。特定のプロパティがメッセージで定義されていない場合は、アダプターはその属性の値を CxBlank に設定します。

**同期イベント通知:** 同期イベント処理の場合は、アダプターはイベントを通知し、統合ブローカーからの応答を待った後、アプリケーションに応答メッセージを送信します。ビジネス・データに対する変更は、戻される応答メッセージに反映されます。イベントを通知する前に、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を設定します。動的メタオブジェクトに設定される値は、以下の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の他の読み取り専用属性は無視されます)。

- **CorrelationID** 動的メタオブジェクトが属性 CorrelationId を含む場合は、発信元アプリケーションが必要とす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と元の要求を突き合わせます。CorrelationID が予期しない値または無効値の場合は、問題が発生します。これは、この属性を使用する前にアプリケーションが関連する要求および応答メッセージを処理する方法を判別するのに役立ちます。同期要求で CorrelationID を設定するには 4 つの方法があります。

1. 値を変更しない。応答メッセージの CorrelationID は、要求メッセージの CorrelationID と同じになります。これは、WebSphere MQ オプション MQRO\_PASS\_CORREL\_ID と同等です。
2. 値を CxIgnore に変更する。コネクターは、デフォルトで要求のメッセージ 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 と同等です。
3. 値を CxBlank に変更する。コネクターは応答メッセージの CorrelationID を設定しません。
4. 値をカスタム値に変更する。これを行うには、応答を処理するアプリケーションでカスタム値を認識する必要があります。

メタオブジェクトで属性 CorrelationID を定義しない場合は、コネクターは自動的に CorrelationID を処理します。

- **ReplyToQueue** 属性 ReplyToQueue に別のキューを指定することによって動的メタオブジェクトを更新する場合は、コネクターは、応答メッセージを指定のキューに送信します。これはお勧めしません。コネクターに対して応答メッセージを別のキューに送信させると、応答メッセージで特定の応答キューを設定するアプリケーションはそのキューで応答を待つと想定されるため、通信に干渉する場合があります。
- **JMS プロパティ** 更新されたビジネス・オブジェクトがコネクターに戻ったときに JMS プロパティ属性用として動的メタオブジェクトに設定された値は、応答メッセージに設定されます。



**非同期要求の処理:** コネクタは、動的メタオブジェクト (存在する場合) を使用して要求メッセージを設定してから発行します。コネクタは、以下のステップを実行してから要求メッセージを送信します。

1. 属性 `CorrelationID` が動的メタオブジェクトに存在する場合は、コネクタは、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で指定されている場合は、コネクタは、応答メッセージでこのキューを渡し、このキューで応答を待ちます。これにより、コネクタ構成プロパティで指定されている `ReplyToQueue` 値をオーバーライドできます。さらに負の `ResponseTimeout` (コネクタが応答を待たないことを示す) を指定した場合は、コネクタは実際には応答を待ちませんが、応答メッセージで `ReplyToQueue` が設定されます。
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、コネクタが終了します。
5. 動的メタオブジェクトで属性 `JMSProperties` を指定した場合は、コネクタによって送信されるアウトバウンド・メッセージに、子動的メタオブジェクトで指定された対応する JMS プロパティが設定されます。

**注:** 動的メタオブジェクトのヘッダー属性が定義されていない場合または `CxIgnore` を指定した場合は、コネクタはデフォルト設定に従います。

**同期要求の処理:** コネクタは、動的メタオブジェクト (存在する場合) を使用して要求メッセージを設定してから発行します。動的メタオブジェクトがヘッダー属性を含む場合は、コネクタは、応答メッセージで検出された対応する新しい値をそのヘッダー属性に設定します。コネクタは、応答メッセージを受信した後、(メタオブジェクトにトランスポート関連のデータを設定するほかに) 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に存在する場合は、アダプターは、応答メッセージで指定された `JMSCorrelationID` でこの属性を更新します。
2. 属性 `ReplyToQueue` が動的メタオブジェクトで定義されている場合は、アダプターは、応答メッセージで指定された `JMSReplyTo` の名前がこの属性を更新します。
3. 属性 `DeliveryMode` が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの `JMSDeliveryMode` ヘッダー・フィールドの値でこの属性を更新します。
4. 属性 `Priority` が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの `JMSPriority` ヘッダー・フィールドの値でこの属性を更新します。
5. 属性 `Destination` が動的メタオブジェクトで定義されている場合は、アダプターは、応答メッセージで指定された `JMSDestination` の名前がこの属性を更新します。

6. 属性 Expiration が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSExpiration ヘッダー・フィールドの値でこの属性を更新します。
7. 属性 MessageID が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSMessageID ヘッダー・フィールドの値でこの属性を更新します。
8. 属性 Redelivered が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSRedelivered ヘッダー・フィールドの値でこの属性を更新します。
9. 属性 TimeStamp が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSTimeStamp ヘッダー・フィールドの値でこの属性を更新します。
10. 属性 Type が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSType ヘッダー・フィールドの値でこの属性を更新します。
11. 属性 UserID が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSXUserID ヘッダー・フィールドの値でこの属性を更新します。
12. 属性 AppID が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSXAppID プロパティー・フィールドの値でこの属性を更新します。
13. 属性 DeliveryCount が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSXDeliveryCount ヘッダー・フィールドの値でこの属性を更新します。
14. 属性 GroupID が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSXGroupID ヘッダー・フィールドの値でこの属性を更新します。
15. 属性 GroupSeq が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSXGroupSeq ヘッダー・フィールドの値でこの属性を更新します。
16. 属性 JMSProperties が動的メタオブジェクトで定義されている場合は、アダプターは、子オブジェクトで定義されているすべてのプロパティーを、応答メッセージで検出された値で更新します。子オブジェクトで定義されているプロパティーがメッセージ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で設定された 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は影響しません。アダプターは、デフォルトですべての応答メッセージの CorrelationID がアダプターによって送信された要求のメッセージ ID に等しいことを要求します。

**エラー処理:** JMS プロパティーをメッセージから読み取れない場合、またはメッセージに書き込めない場合は、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 が存在しないか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。いずれの場合も、ログに記録されたメッセージはコネクターのメッセージ・ファイルからのものです。

---

## 始動ファイルの構成

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

### Windows

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

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

### UNIX

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

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

---

## 始動

### コネクタの始動

コネクタは、**コネクタ始動スクリプト**を使用して明示的に開始する必要があります。始動スクリプトは、コネクタのランタイム・ディレクトリー、

`ProductDir¥connectors¥connName`

にある必要があります。このディレクトリー名の `connName` はコネクタを表しています。表 3 が示すとおり、始動スクリプトの名前はオペレーティング・システム・プラットフォームにより異なります。

表 3. コネクタの始動スクリプト

オペレーティング・システム	始動スクリプト
UNIX 系システム	<code>connector_manager_connName</code>
Windows	<code>start_connName.bat</code>

以下のいずれかの方法で、コネクタ始動スクリプトを起動できます。

- Windows システムでは、「スタート」メニューから、「プログラム」>「IBM WebSphere Business Integration Adapters」>「Adapters」>「Connectors」を選択する。デフォルトでは、プログラム名は“IBM WebSphere Business Integration Adapters”ですが、カスタマイズもできます。または、コネクタへのデスクトップ・ショートカットを作成することもできます。
- コマンド行から起動する。

- Windows システム:  
`start_connName connName brokerName [-cconfigFile ]`
- UNIX 系システム:  
`connector_manager_connName -start`

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

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

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

- Adapter Monitor から起動する (WebSphere Business Integration Adapters 製品のみ)。System Manager を始動すると起動されます。  
このツールを使用して、コネクタをロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除できます。
- System Monitor から起動する (WebSphere InterChange Server 製品のみ)  
このツールを使用して、コネクタをロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除できます。
- Windows システムでは、コネクタを Windows のサービスとして開始されるように構成できる。この場合、自動サービスでは Windows システムがブートするとき、手動サービスでは「Windows サービス」ウィンドウからサービスを開始するとき、コネクタが開始されます。

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

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

---

## コネクタの停止

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

- コネクタの始動スクリプトを使用してコネクタをコマンド行から始動した場合:

- Windows システムでは、始動スクリプトを起動すると、そのコネクタ用の個別の「コンソール」ウィンドウが作成されます。このウィンドウで、「Q」と入力して Enter キーを押すと、コネクタが停止します。
- UNIX 系システムではコネクタはバックグラウンドで稼働するため、個別のウィンドウはありません。その代わりに、以下のコマンドを実行してコネクタを停止します。

```
connector_manager_connName -stop
```

ここで、*connName* はコネクタ名です。

- System Manager を始動したときに起動された Adapter Monitor から起動した場合 (WebSphere Business Integration Adapters 製品のみ):  
このツールを使用して、コネクタをロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除できます。
- System Monitor から起動した場合 (WebSphere InterChange Server 製品のみ):  
このツールを使用して、コネクタをロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除できます。
- Windows システムでは、コネクタを Windows のサービスとして始動するように構成できます。この場合、Windows システムがシャットダウンされるとコネクタは停止します。



---

## 第 3 章 ビジネス・オブジェクトの作成および変更

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

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

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

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

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

---

### アダプターのビジネス・オブジェクトの構造

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

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

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

## ビジネス・オブジェクト・プロパティのサンプル

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

```
[ReposCopy]
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = Sample_WebSphere MQ_LegacyContact
Version = 1.0.0
```

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

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

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

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

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



```

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

```

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

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

---

## エラー処理

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

Message number  
Message text

コネクタは、以降のセクションで説明する方法で特定のエラーを処理します。

### アプリケーション・タイムアウト

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

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

### アンサブスクライブされたビジネス・オブジェクト

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

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

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

## コネクタがアクティブでない

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

## データ・ハンドラーによる変換

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の動作を詳細にトレースできます。デフォルトでは、トレース・メッセージは `STDOUT` に書き込まれます。トレース・メッセージの構成の詳細については、21 ページの『第 2 章 アダプターのインストールおよび構成』のコネクタ構成プロパティの説明を参照してください。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、「コネクタ開発ガイド」を参照してください。

コネクタのトレース・メッセージに推奨される内容を以下に示します。

- レベル 0      コネクタのバージョンを確認するトレース・メッセージに使用します。
- レベル 1      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 `gotAppEvent()` または `executeCollaboration()` から `InterChange Server` に送付されるたびに記録されるトレース・メッセージに使用します。
- レベル 3      メッセージからビジネス・オブジェクトへの変換およびビジネス・オブジェクトからメッセージへの変換に関する情報を提供するトレース・メッセージや、出力キューへのメッセージのデリバリーに関する情報を提供するトレース・メッセージに使用します。

- レベル 4      コネクターが動作を開始または終了した時間を記録するトレース・メッセージに使用します。
- レベル 5      コネクターの初期化を示すトレース・メッセージ、アプリケーション内で実行されるステートメントを表すトレース・メッセージ、メッセージが除去されるかキューに送出されるたびに記録されるトレース・メッセージ、または、ビジネス・オブジェクトのダンプを記録するトレース・メッセージに使用します。

---

## 第 4 章 トラブルシューティング

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

---

### 始動時の問題

#### 問題

初期化中にコネクタが予期しないエラーでシャットダウンし、次のメッセージが報告されました: Exception in thread "main" java.lang.NoClassDefFoundError: javax/jms/JMSException...

初期化中にコネクタが予期しないエラーでシャットダウンし、次のメッセージが報告されました: Exception in thread "main" java.lang.NoClassDefFoundError: com/ibm/mq/jms/MQConnectionFactory...

初期化中にコネクタが予期しないエラーでシャットダウンし、次のメッセージが報告されました: Exception in thread "main" java.lang.NoClassDefFoundError: javax/naming/Referenceable...

初期化中にコネクタ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

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

#### 考えられる処置/説明

コネクタは、IBM WebSphere MQ Java クライアント・ライブラリーからファイル `.jms.jar` を見つけることができません。start\_connector.bat 内の変数 `WebSphereMQ_JAVA_LIB` が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認します。

コネクタは、IBM WebSphere MQ Java クライアント・ライブラリーからファイル `com.ibm.mqjms.jar` を見つけることができません。start\_connector.bat 内の変数 `WebSphereMQ_JAVA_LIB` が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認します。

コネクタは、IBM WebSphere MQ Java クライアント・ライブラリーからファイル `jndi.jar` を見つけることができません。start\_connector.bat 内の変数 `WebSphereMQ_JAVA_LIB` が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認します。

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

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

---

### イベント処理

#### 問題

コネクタは、MQRFH2 ヘッダーを持つすべてのメッセージをデリバリーします。

#### 考えられる処置/説明

MQMD WebSphere MQ ヘッダーを持つメッセージのみをデリバリーするには、出力キューの URI の名前に `?targetClient=1` を付加します。例えば、メッセージをキュー `queue://my.queue.manager/OUT` に出力する場合は、URI を

`queue://my.queue.manager/OUT?targetClient=1` に変更します。詳細については、21 ページの『第 2 章 アダプターのインストールおよび構成』を参照してください。

---

**問題**

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

**考えられる処置/説明**

これは WebSphere MQ MQMD メッセージ・ヘッダーの制限であり、コネクタの制限ではありません。

---

---

## 付録 A. コネクターの標準構成プロパティ

この付録では、WebSphere Business Integration アダプターのコネクタ・コンポーネントの標準構成プロパティについて説明しています。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を対象としています。

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

コネクタによっては、一部の標準プロパティが使用されないことがあります。Connector Configurator から統合ブローカーを選択するときには、そのブローカーで稼働するアダプターのために構成が必要な標準プロパティのリストが表示されます。

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

**注:** 本書では、ディレクトリー・パスの規則として円記号 (¥) 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。

---

### 新規プロパティと削除されたプロパティ

本リリースには、次の標準プロパティが追加されました。

#### 新規プロパティ

- XMLNamespaceFormat

#### 削除されたプロパティ

- RestartCount
- RHF2MessageDomain

---

### 標準コネクタ・プロパティの構成

アダプター・コネクタには 2 つのタイプの構成プロパティがあります。

- 標準構成プロパティ
- コネクタ固有のプロパティ

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

## Connector Configurator の使用

Connector Configurator からコネクタ・プロパティを構成します。Connector Configurator には、System Manager からアクセスします。Connector Configurator の使用法の詳細については、付録の『Connector Configurator』を参照してください。

**注:** Connector Configurator と System Manager は、Windows システム上でのみ動作します。コネクタを UNIX システム上で稼働している場合でも、これらのツールがインストールされた Windows マシンが必要です。UNIX 上で動作するコネクタのコネクタ・プロパティを設定する場合は、Windows マシン上で System Manager を起動し、UNIX の統合ブローカーに接続してから、コネクタ一用の Connector Configurator を開く必要があります。

## プロパティ値の設定と更新

プロパティ・フィールドのデフォルトの長さは 255 文字です。

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

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

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

### • 動的

変更を System Manager に保管すると、変更が即時に有効になります。コネクタがスタンドアロン・モードで (System Manager から独立して) 作動している場合 (例えば、WebSphere Message Brokers の 1 つで作動している場合)、構成ファイルによるプロパティの変更のみが可能です。この場合、動的更新は実行できません。

### • コンポーネント再始動

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

### • サーバー再始動

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

### • エージェント再始動 (ICS のみ)

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

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



## 標準プロパティの要約

表 4 は、標準コネクタ構成プロパティの早見表です。すべてのコネクタがこれらのすべてのプロパティを使用しているわけではありません。標準プロパティの依存関係は RepositoryDirectory に基づいているため、プロパティの設定は統合ブローカーごとに異なります。

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

表 4. 標準構成プロパティの要約

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

表 4. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
DuplicateEventElimination	True または False	False	コンポーネント再始動	JMS トランスポートのみ: Container Managed Events は <NONE> でなければならない
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
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

表 4. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならぬ
MessageFileName	パスまたはファイル名	InterchangeSystem.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	CONNECTORNAME/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は True でなければならぬ
OADAutoRestartAgent	True または False	False	動的	Repository Directory は <REMOTE> でなければならぬ
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければならぬ
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければならぬ
PollEndTime	HH:MM	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒)  no (ポーリングを使用不可にする)  key (コネクタのコマンド・プロンプト・ウィンドウで文字 p が入力された場合にのみポーリングする)	10000	動的	
PollQuantity	1 から 500	1	エージェント再始動	JMS トランスポートのみ: Container Managed Events が指定されている
PollStartTime	HH:MM(HH は 0 から 23、MM は 0 から 59)	HH:MM	コンポーネント再始動	
RepositoryDirectory	メタデータ・リポジトリの場所		エージェント再始動	ICS の場合は <REMOTE> に設定する。 WebSphere MQ Message Brokers および WAS: C:¥crossworlds¥リポジトリに設定

表 4. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
RequestQueue	有効な JMS キュー名	CONNECTORNAME/REQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
ResponseQueue	有効な JMS キュー名	CONNECTORNAME/RESPONSEQUEUE	コンポーネント再始動	Delivery Transport は JMS: Repository Directory が <REMOTE> のときのみ必要
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
WireFormat	CwXML、CwBO	CwXML	エージェント再始動	Repository Directory が <REMOTE> でない場合は CwXML。Repository Directory が <REMOTE> の場合は CwBO
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

このセクションでは、各標準コネクタ構成プロパティの定義を示します。

### AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

## AdminOutQueue

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

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

## AgentConnections

`RepositoryDirectory` が `<REMOTE>` の場合のみ適用できます。

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

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

## AgentTraceLevel

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

## ApplicationName

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

## BrokerType

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

## CharacterEncoding

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

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

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

## ConcurrentEventTriggeredFlows

`RepositoryDirectory` が `<REMOTE>` の場合のみ適用できます。

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 構成プロパティに、1 より大きい値を設定します。

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

## ContainerManagedEvents

このプロパティにより、JMS イベント・ストアを使用する JMS 対応コネクターが、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も構成する必要があります。

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

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

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

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

## ControllerStoreAndForwardMode

RepositoryDirectory が <REMOTE> の場合のみ適用できます。

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

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

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

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

デフォルト値は true です。

## ControllerTraceLevel

RepositoryDirectory が <REMOTE> の場合のみ適用できます。

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

## DeliveryQueue

DeliveryTransport が JMS の場合のみ適用できます。

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

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

## DeliveryTransport

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

- ICS がブローカー・タイプの場合は、DeliveryTransport プロパティの指定可能な値は MQ、IDL、または JMS であり、デフォルトは IDL になります。

- `RepositoryDirectory` がローカル・ディレクトリーの場合、指定可能な値は `JMS` のみです。

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

## WebSphere MQ および IDL

イベントのデリバリー・トランスポートには、IDL ではなく WebSphere MQ を使用してください (1 種類の製品のみを使用する必要がある場合を除きます)。

WebSphere MQ が IDL よりも優れている点は以下のとおりです。

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

## JMS

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

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

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

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

この環境では、WebSphere MQ クライアント内でメモリが使用されるため、(サーバー側の) コネクタ・コントローラーと (クライアント側の) コネクタの両方を



始動するのは困難な場合があります。ご使用のシステムのプロセス・ヒープ・サイズが 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 プロパティの値を 11 または 12 に設定する。このプロパティの詳細については、「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。このプロパティの値はオプションです。

デフォルトはありません。

## jms.UserName

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

デフォルトはありません。

## ListenerConcurrency

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

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

## Locale

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

ロケール名は、次の書式で指定します。

`ll_TT.codeset`

ここで、以下のように説明されます。

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

デフォルトでは、ドロップ・リストには、サポートされるロケールの一部のみが表示されます。ドロップ・リストに、サポートされる他の値を追加するには、製品ディレクトリーにある `¥Data¥Std¥stdConnProps.xml` ファイルを手動で変更する必要があります。詳細については、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。コネクタがグローバル化に対応していない場合、このプロパティの有効な値は `en_US` のみです。特定のコネクタがグローバル化に対応しているかどうかを判別するには、以下の Web サイトにあるコネクタのバージョン・リストを参照してください。

<http://www.ibm.com/software/websphere/wbiadapters/infocenter>、または  
<http://www.ibm.com/websphere/integration/wicsserver/infocenter>

## LogAtInterchangeEnd

`RepositoryDirectory` が `<REMOTE>` の場合のみ適用できます。

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

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

## MaxEventCapacity

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

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。

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

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

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。`OADAutoRestartAgent` プロパティーを有効にするには、値を `true` に設定する必要があります。

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` の再試行間隔の分数を指定します。コネクター・エージェントがこの再試行間隔の間に再始動しないと、コネクター・コントローラ

ーが OAD にコネクタ・エージェントの再始動を再度要求します。OAD はこの再試行処理を `OADMaxNumRetry` プロパティで指定されている回数だけ繰り返します。OADAutoRestartAgent プロパティを有効にするには、値を `true` に設定する必要があります。

デフォルト値は 10 です。

## PollEndTime

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

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

## PollFrequency

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

- ポーリング・アクション間のミリ秒数。
- ワード `key`。コネクタは、コネクタのコマンド・プロンプト・ウィンドウで文字 `p` が入力されたとき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`。コネクタはポーリングを実行しません。このワードは小文字で入力します。

デフォルト値は 10000 です。

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

## PollQuantity

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

## PollStartTime

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

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

## RequestQueue

統合ブローカーが、ビジネス・オブジェクトをコネクタ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

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

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

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

## ResponseQueue

`DeliveryTransport` が JMS の場合のみ適用でき、`RepositoryDirectory` が `<REMOTE>` の場合のみ必要です。

JMS 応答キューを指定します。JMS 応答キューは、応答メッセージをコネクタ・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

コネクタによるコネクタ自体の再始動の試行回数を指定します。このプロパティを並列コネクタに対して使用する場合、コネクタのマスター側のアプリケーション固有のコンポーネントが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

コネクタによるコネクタ自体の再始動の試行間隔を分単位で指定します。このプロパティを並列コネクタに対して使用する場合、コネクタのマスター側のアプリケーション固有のコンポーネントが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。指定できる値の範囲は 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できます。

JMS イベント・ストアを使用する JMS 対応コネクタでの保証付きイベント・デリバリーをサポートするコネクタ・フレームワークに、JMS ソース・キューを指定します。詳細については、68 ページの『`ContainerManagedEvents`』を参照してください。

デフォルト値は `CONNECTOR/SOURCEQUEUE` です。

## SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用できます

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

デフォルト値は CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

## SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用できます。

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

デフォルト値は CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

## SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用できます。

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

デフォルト値は 0 です。

## WireFormat

トランスポートのメッセージ・フォーマットです。

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

## WsifSynchronousRequest Timeout

WAS 統合ブローカーでのみ使用されます。

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

デフォルト値は 0 です。

## XMLNameSpaceFormat

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

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

デフォルト値は short です。



---

## 付録 B. Connector Configurator

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

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

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

注:

本書では、ディレクトリー・パスの規則として円記号 (¥) 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。

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

---

### Connector Configurator の概要

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

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

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

- コネクタを構成するためのコネクタ固有のプロパティ・テンプレートを作成します。
- **コネクタ構成ファイル**を作成します。インストールするコネクタごとに 1 つ構成ファイルを作成する必要があります。
- 構成ファイルのプロパティを設定します。  
場合によっては、コネクタ・テンプレートでプロパティに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、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 (80 ページの『スタンドアロン・モードでの Configurator の実行』を参照)。

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

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

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

ただし**コネクタ固有プロパティ**の場合は、最初にプロパティを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクタ固有プロパティのテンプレートを作成します。システム内で既にテンプレートが作成されている場合には、作成されているテンプレートを使用します。システム内でまだテンプレートが作成されていない場合には、81 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

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

---

## Connector Configurator の始動

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

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

### スタンドアロン・モードでの Configurator の実行

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

これを行うには、以下のステップを実行します。

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

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

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存してください (86 ページの『構成ファイルの完成』を参照)。

---

## System Manager からの Configurator の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

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

---

## コネクタ固有のプロパティ・テンプレートの作成

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

コネクタ固有プロパティのテンプレートを新規に作成するか、または既存の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、81 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でのこのテンプレートを保管します。

### 新規テンプレートの作成

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

テンプレートは以下のように作成します。

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

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

### 一般特性の指定

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

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

プロパティの一般特性の選択を終えたら、「値」タブをクリックします。

### 値の指定

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

1. 「値」タブをクリックします。「一般」のパネルに代わって「値」の表示パネルが表示されます。
2. 「プロパティを編集」表示でプロパティの名前を選択します。
3. 「最大長」および「最大複数値」のフィールドで、変更を行ってください。次のステップで説明するように、プロパティの「プロパティ値」ダイアログ・ボックスを開かない限り、そのプロパティ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. 値テーブルの左隅上にあるボックスを右マウス・ボタン・クリックして、「追加」をクリックします。「プロパティ値」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のタイプに応じて、値のみ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、「プロパティ値」ダイアログ・ボックスで入力した値と、作成した以前の値が表示されます。
  - 「デフォルト値」の列では、値のいずれかをデフォルトとして指定することができます。
  - 「値の範囲」の列には、「プロパティ値」ダイアログ・ボックスで入力した範囲が表示されます。

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

## 依存関係の設定

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

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

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

>= (より大か等しい)

<= (より小か等しい)

4. 「条件値」フィールドで、依存プロパティをテンプレートに組み込むために必要な値を入力します。
5. 「使用可能なプロパティ」表示で強調表示された依存プロパティで、矢印をクリックし、「依存プロパティ」表示に移動させます。
6. 「完了 (Finish)」をクリックします。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。
- 「統合ブローカー」フィールドで、ICS、WebSphere Message Brokers、WAS のいずれかの接続を選択します。
- 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と一貫性をもつ一意の名前である必要があります。

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

- **システム接続**

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

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

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

使用するテンプレートを選択し、「OK」をクリックします。

3. 構成しているコネクタの構成画面が表示されます。タイトル・バーに、統合ブローカーとコネクタの名前が表示されます。ここですべてのフィールドに値を入力して定義を完了するか、ファイルを保管して後でフィールドに値を入力するかを選択できます。
  4. ファイルを保管するには、「ファイル」>「保管」>「ファイルに」をクリックするか、「ファイル」>「保管」>「プロジェクトに」をクリックします。プロジェクトに保管するには、System Manager が実行中である必要があります。ファイルとして保管する場合は、「ファイル・コネクタを保管」ダイアログ・ボックスが表示されます。\*.cfg をファイル・タイプとして選択し、「ファイル名」フィールド内に名前が正しいスペル (大文字と小文字の区別を含む) で表示されていることを確認してから、ファイルを保管するディレクトリーにナビゲートし、「保管」をクリックします。Connector Configurator のメッセージ・パネルの状況表示に、構成ファイルが正常に作成されたことが示されます。
- 重要:** ここで設定するディレクトリー・パスおよび名前は、コネクタの始動ファイルで指定するコネクタ構成ファイルのパスおよび名前に一致している必要があります。
5. この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、コネクタ定義を完了します。

---

## 既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

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

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

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

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

1. Connector Configurator 内で、「ファイル」>「開く」>「ファイルから」とクリックします。
2. 「ファイル・コネクタを開く」ダイアログ・ボックス内で、以下のいずれかのファイル・タイプを選択して、使用可能なファイルを調べます。
  - 構成 (\*.cfg)
  - ICS リポジトリー (\*.in、\*.out)  
ICS 環境でのコネクタ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定義が含まれている場合は、ファイルを開くとすべての定義が表示されます。
  - すべてのファイル (\*.\*)  
コネクタのアダプター・パッケージに \*.txt ファイルが付属していた場合、または別の拡張子で定義ファイルが使用可能である場合は、このオプションを選択します。
3. ディレクトリー表示内で、適切なコネクタ定義ファイルへ移動し、ファイルを選択し、「開く」をクリックします。

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

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

---

## 構成ファイルの完成

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

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



1. 「標準のプロパティ」タブで、BrokerType プロパティの値フィールドを選択します。ドロップダウン・メニューで、値 ICS、WMQI、または WAS を選択します。
2. 選択したブローカーに関連付けられているプロパティが「標準のプロパティ」タブに表示されます。ここでファイルを保管するか、または 90 ページの『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。
3. 構成が完了したら、「ファイル」>「保管」>「プロジェクトに」を選択するか、または「ファイル」>「保管」>「ファイルに」を選択します。

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

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

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

---

## 構成ファイル・プロパティの設定

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

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

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

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

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

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

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

標準プロパティとコネクタ固有プロパティの違いは、以下のとおりです。

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

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

- 背景がグレーのフィールドは、標準のプロパティを表します。値を変更することはできますが、名前の変更およびプロパティの除去はできません。
- 背景が白のフィールドは、アプリケーション固有のプロパティを表します。これらのプロパティは、アプリケーションまたはコネクタの特定のニーズによって異なります。値の変更も、これらのプロパティの除去も可能です。
- 「値」フィールドは構成できます。
- 「更新メソッド」フィールドは通知を行うもので、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・プロパティの設定

標準のプロパティの値を変更するには、以下の手順を実行します。

1. 値を設定するフィールド内でクリックします。
2. 値を入力するか、ドロップダウン・メニューが表示される場合にはメニューから値を選択します。
3. 標準のプロパティの値をすべて入力すると、以下のいずれかを実行することができます。
  - 変更内容を破棄し、元の値を保持したままで Connector Configurator 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 内の他のカテゴリの値を入力するには、そのカテゴリのタブを選択します。「標準のプロパティ」 (またはその他のカテゴリ

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

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

## アプリケーション固有の構成プロパティの設定

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

1. グリッドの左上端の部分で右マウス・ボタン・クリックします。ポップアップ・メニュー・バーが表示されます。「追加」をクリックしてプロパティを追加します。子プロパティを追加するには、親行番号を右マウス・ボタン・クリックして、「子を追加」をクリックします。
2. プロパティまたは子プロパティの値を入力します。
3. プロパティを暗号化するには、「暗号化」ボックスを選択します。
4. 88 ページの『標準コネクタ・プロパティの設定』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。

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

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

### コネクタ・プロパティの暗号化

アプリケーション固有のプロパティは、「プロパティを編集」ウィンドウの「暗号化」チェック・ボックスにチェックマークを付けると、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。

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

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

## 更新メソッド

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

## サポートされるビジネス・オブジェクト定義の指定

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

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

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

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

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

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

ほとんどのコネクターの場合、選択可能な項目は「最大限の努力」のみです。

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

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

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

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

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

### ご使用のブローカーが WAS の場合

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

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

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

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

## 関連付けられたマップ (ICS のみ)

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

ビジネス・オブジェクトのリストには、エージェントでサポートされるアプリケーション固有のビジネス・オブジェクトと、コントローラーがサブスクライブ・コラボレーションに送信する、対応する汎用オブジェクトが含まれます。マップの関連によって、アプリケーション固有のビジネス・オブジェクトを汎用ビジネス・オブジェクトに変換したり、汎用ビジネス・オブジェクトをアプリケーション固有のビジネス・オブジェクトに変換したりするときに、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、関連付けられたマップを明示的にバインドすることが必要になります。

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

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

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

以下の手順を実行して、マップを明示的にバインドします。

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

## リソース (ICS)

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

すべてのコネクタ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・エージェントを実行している場合、この機能を使用することはお勧めできません。

## メッセージング (ICS)

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

## トレース/ログ・ファイル値の設定

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

ログとトレースの値を変更するには、以下の手順を実行します。

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

**注:** STDOUT オプションは、Windows プラットフォームで実行しているコネクタの「トレース/ログ・ファイル」タブでのみ使用できます。

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

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

## データ・ハンドラー

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

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

---

## 構成ファイルの保管

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

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

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

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・プロパティと同様に他の構成プロパティも変更する必要があります。

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

- Connector Configurator で既存の構成ファイルを開きます。
- 「標準のプロパティ」タブを選択します。
- 「標準のプロパティ」タブの「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、プロパティ画面の利用可能なタブおよびフィールド選択がただちに更新され、選択した新規ブローカーに適したタブとフィールドのみが表示されます。

---

## 構成の完了

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

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

---

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

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

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

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

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

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

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

---

## 付録 C. チュートリアル

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

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

表記規則のガイドについては、この文書のまえがきを参照してください。

---

### チュートリアルについて

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

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

始動した Adapter for WebSphere MQ は、ApplicationX が入力キューに書き込んだ連絡メッセージを検索します。アダプターは区切りデータ・ハンドラーを使用することにより、これらのメッセージを `Sample_WebSphereMQ_LegacyContact` ビジネス・オブジェクトに変換し、統合ブローカーに送達します。テスト・コネクタ (やはり、WBI をインストールすると組み込まれているコンポーネント) を使用することにより、ポート・コネクタをシミュレートし、Adapter for WebSphere MQ が発行したビジネス・オブジェクトを検索し、属性を確認することができます。データを変更してから、メッセージを統合ブローカーに再送達します。ここからメッセージは Adapter for WebSphere MQ に送信され、メッセージに変換され、アダプターの出力キュー (ApplicationX の入力キュー) に送達されます。このチュートリアルでは、アダプターは WebSphere MQ Integrator Broker 用に構成されていますが、チュートリアルを実行するためにこのブローカーを実際にインストールおよび構成する必要はありません。

---

## 始める前に

チュートリアルを開始する前に、以下のことを確認してください。

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

---

## 環境のセットアップ

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

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

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

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

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

- DEFINE QL('PortConnector/FAULTQUEUE')
- DEFINE QL('PortConnector/REQUESTQUEUE')
- DEFINE QL('PortConnector/RESPONSEQUEUE')
- DEFINE QL('PortConnector/SYNCHRONOUSREQUESTQUEUE')
- DEFINE QL('PortConnector/SYNCHRONOUSRESPONSEQUEUE')

2. **アダプターの構成** Connector Configurator を使用して、*sample\_folder*\WebSphereMQConnector.cfg を開きます。Connector Configurator の使用法に関する詳細情報は、79 ページの『付録 B. Connector Configurator』を参照してください。コネクタ固有プロパティの詳細については、24 ページの『コネクタ固有のプロパティ』を参照してください。アダプターをまだ構成していない場合は、ご使用のシステムに応じた「インストール・ガイド」の説明に従って構成してください。さらに、以下に示す値と一致するようにアダプターの構成プロパティを確認または変更します。

- Broker Type このプロパティを WMQI に設定します。
- Repository Directory このプロパティを *sample\_folder* ディレクトリーに設定します。

以下のコネクタ固有プロパティを設定します。

- ConfigurationMetaObject このプロパティを Sample\_WebSphereMQ\_MO\_Config に設定します。
- DataHandlerConfigMO このプロパティを Sample\_WebSphereMQ\_MO\_DataHandler に設定します。
- DataHandlerMimeType このプロパティを text/delimited に設定します。
- DataHandlerClassName このプロパティを com.crossworlds.DataHandlers.text.delimited に設定します。
- ErrorQueue このプロパティを queue://crossworlds.queue.manager/MQCONN.ERROR に設定します。
- InProgressQueue このプロパティを queue://crossworlds.queue.manager/MQCONN.IN\_PROGRESS に設定します。
- InputQueue このプロパティを queue://crossworlds.queue.manager/MQCONN.IN に設定します。
- hostname このプロパティをご使用のマシン名に設定します。
- port このプロパティを 1414 に設定します。
- channel このプロパティを CHANNEL1 に設定します。
- UnsubscribedQueue このプロパティを queue://crossworlds.queue.manager/MQCONN.UNSUBSCRIBED に設定します。

3. **ポート・コネクターの構成** Connector Configurator を使用して、以下の標準プロパティを設定します。

- Broker Type このプロパティを WMQI に設定します。
- Repository Directory このプロパティを *sample\_folder* ディレクトリーに設定します。
- RequestQueue このプロパティを WebSphereMQConnector/DELIVERYQUEUE に設定します (WebSphere MQ アダプターの DeliveryQueue プロパティ値)。

- `DeliveryQueue` このプロパティを `WebSphereMQConnector/REQUESTQUEUE` に設定します (`WebSphere MQ` アダプターの `RequestQueue` プロパティ値)。
4. **ビジネス・オブジェクトのサポート** ビジネス・オブジェクトを使用するには、まずアダプターがビジネス・オブジェクトをサポートする必要があります。  
Connector Configurator を使用して、`WebSphere MQ` アダプターの「サポートされているビジネス・オブジェクト」タブをクリックして、表 5 に記載されているビジネス・オブジェクトを追加します。「メッセージ・セット ID」をサポートされているそれぞれのビジネス・オブジェクトごとに固有の値に設定します。

表 5. JMS アダプターについてサポートされるサンプル・ビジネス・オブジェクト

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

Connector Configurator を使用して、`sample_folder` 内にあるポート・コネクタ定義 `PortConnector.cfg` を開きます。次に、表 6 に記載されたサポートされるビジネス・オブジェクトとメッセージ・セット ID を追加します。

表 6. ポート・コネクタについてサポートされるサンプル・ビジネス・オブジェクト

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

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

6. **コネクタ開始スクリプトの作成または更新**

#### Windows の場合

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

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

#### UNIX の場合

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

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

## シナリオの実行

シナリオを実行する前に、以下の手順を実行します。

1. **Adapter for `WebSphere MQ`** がまだ稼働していない場合は始動します。
2. **Visual テスト・コネクタ** がまだ稼働していない場合は始動します。

## 静的メタオブジェクトを使用したシナリオ

チュートリアルはこのセクションでは、静的メタオブジェクトを使用したシナリオについて説明します。静的メタオブジェクトの詳細については、32 ページの『静的メタオブジェクト』を参照してください。

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

ターがメッセージを検索し、これらのメッセージを Sample\_WebSphereMQ\_LegacyContact ビジネス・オブジェクトに変換し、さらに、逆に Sample\_WebSphereMQ\_LegacyContact ビジネス・オブジェクトを連絡メッセージに変換することが可能になります。

## 動的メタオブジェクトを使用したシナリオ

このシナリオでは、動的メタオブジェクトを使用して、静的メタオブジェクトのシナリオに定義された各種のキューにビジネス・オブジェクトを転送する方法について説明します。動的メタオブジェクトの詳細については、39 ページの『動的子メタオブジェクト』を参照してください。このシナリオの前提条件については、98 ページの『始める前に』を参照してください。さらに、101 ページの『静的メタオブジェクトを使用したシナリオ』の説明に従ってポート・コネクターをインストールおよび構成する必要があります。以下のステップでは、Sample\_WebSphereMQ\_LegacyContact の子メタオブジェクトの属性を作成します。特に、この子メタオブジェクトの出力キュー値を変更することにより、Sample\_WebSphereMQ\_LegacyContact ビジネス・オブジェクトを新規のキューに転送します。

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

```
DEFINE QL('REROUTE.IN')
```
4. **Adapter for WebSphere MQ** がまだ稼働していない場合は始動します。
5. **Visual テスト・コネクター** がまだ稼働していない場合は始動します。
6. **ポート・コネクターのシミュレート** (静的メタオブジェクトを使用したシナリオでこのステップをすでに実行している場合は、この作業をスキップして次の作業に進んでください。) Visual テスト・コネクターを使用して、ポート・コネクターのプロファイルを定義します。
  - a. 「Visual テスト・コネクター」メニューから「ファイル」->「プロファイルを作成/選択」を選択し、次に、「コネクター・プロファイル」メニューから「ファイル」->「新規プロファイル」を選択します。
  - b. Samples ディレクトリー内にあるポート・コネクター構成ファイル PortConnector.cfg を選択して、Connector Name および Broker Type を構成してから「OK」をクリックします。
  - c. 作成したプロファイルを選択し、「OK」をクリックします。



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



---

## 特記事項

本書に記載の製品、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  
U.S.A.

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

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

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

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

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

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

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

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

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

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

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

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

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

---

## プログラミング・インターフェース情報

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

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

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

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

---

## 商標

以下は、IBM Corporation の商標です。

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

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

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

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。



WebSphere Business Integration Adapter Framework V2.4.0