WebSphere Business Integration Server
Express and Express Plus Adapters

Adapter for JMS ユーザーズ・ガイド

バージョン 4.3.1

WebSphere Business Integration Server
Express and Express Plus Adapters

Adapter for JMS ユーザーズ・ガイド

バージョン 4.3.1

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、 87 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 Server Express バージョン 4.3.1、IBM WebSphere Business Integration Server Express Plus バージョン 4.3.1、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： WebSphere Business Integration Server Express and Express Plus Adapters Adapter for JMS User Guide Version 4.3.1

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 2004.8

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

本書について

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

本書では、IBM WebSphere Business Integration Server Express に付属する Adapter for JMS の構成、ビジネス・オブジェクト開発、およびトラブルシューティングについて説明します。

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

対象読者

本書は、WebSphere Business Integration システムをお客様のサイトでサポートおよび管理する、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 JMS 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。

関連資料

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

関連文書は、http://www.ibm.com/websphere/wbiserverexpress/infocenter でダウンロード、インストール、および表示することができます。

注: 本書の発行後に公開されたテクニカル・サポートの技術情報や速報に、本書の対象製品に関する重要な情報が記載されている場合があります。これらの技術情報や速報は、WebSphere Business Integration のサポート Web サイト (http://www.ibm.com/software/integration/websphere/support/) で参照できます。適

切なコンポーネント領域を選択し、「Technotes (技術情報)」セクションと「Flashes (速報)」セクションを参照してください。

表記上の規則

本書は下記の規則に従って編集されています。

`Courier フォント`	コマンド名、ファイル名、入力情報、システムが画面に出力した情報などのリテラル値を示します。
太字	初出語を示します。
斜体	変数名または相互参照を示します。
青のアウトライン	青のアウトラインは、マニュアルをオンラインで表示するときにのみ見られるもので、相互参照用のハイパーリンクを示します。 アウトラインの内側をクリックすることにより、参照先オブジェクトにジャンプできます。
{ }	構文の記述行の場合、中括弧 {} で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
[ ]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプション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意味します。
< >	命名規則では、不等号括弧は名前の個々の要素を囲み、各要素を区別します。 (例: `<server_name><`*`connector_name`* `>tmp.log`)
ProductDir	IBM WebSphere Business Integration Server Express for Adapters 製品のインストール先ディレクトリーを表します。各プラットフォームのデフォルトは、以下のとおりです。 Windows: IBM¥WebSphereServer OS/400: /QIBM/ProdData/WBIServer43/product Linux: /home/${username}/IBM/WebSphereServer
/、¥	本書では、ディレクトリー・パスに円記号 (¥) を使用します。OS/400 および Linux では、ディレクトリー・パスにスラッシュ (/) を使用します。すべての WebSphere Business Integration Server Express システム製品のパス名は、ご使用のシステムで WebSphere Business Integration システムがインストールされているディレクトリーを基準とした相対パス名です。
UNIX: および Windows:	これらのいずれかで始まる段落は、オペレーティング・システム間の相違を列挙した注記です。
u	この記号は、**UNIX/Windows** の段落の終わりを示します。また、複数段落にわたる注釈の終わりを示します。
%text% および *$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 環境での同等の表記は *$text* です。これは、UNIX 環境変数 *text* の値を示します。

目次

本リリースの新機能

リリース 4.3.1 の新機能

本リリースでは、以下のオペレーティング・システムのサポートが追加されました。

- IBM OS/400 V5R2、V5R3
- Red Hat Enterprise Linux AS 3.0
- SuSE Linux Enterprise Server 8.1 (SP3 を適用)
- Microsoft Windows 2003 (実動モードでの InterChange Server Express およびアダプターのみ)

リリース 4.3 の新機能

本書の最初のリリースです。

第 1 章 Adapter for JMS の概要

Adapter for JMS は、IBM WebSphere Business Integration Server Express のランタイム・コンポーネントです。このアダプターの基本機能は、WebSphere InterChange Server Express が、JMS メッセージの形式でデータの送受信を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るようにするコネクターです。

JMS は、企業メッセージング・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。これは、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送受信できるように設計されています。

コネクターは、アプリケーション固有のコンポーネントおよびコネクター・フレームワークから構成されます。アプリケーション固有のコンポーネントには、特定のアプリケーションに合わせて作成されたコードが含まれます。コネクター・フレームワークは統合ブローカーとアプリケーション固有のコンポーネントの間の仲介役として機能し、そのコードはどのコネクターにも共通です。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントの間で、以下のサービスを提供します。

- ビジネス・オブジェクトの送受信
- 始動メッセージおよび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およびコネクター・フレームワークについて説明します。ここでは、これらの両方のコンポーネントを「コネクター」と呼んでいます。

注: すべての WebSphere Business Integration Server Express および Express Plus アダプターは、WebSphere InterChange Server Express Integration Broker と動作します。WebSphere InterChange Server Express Integration Broker については、「*システム・インプリメンテーション・ガイド*」を参照してください。

Adapter for JMS の環境

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

アダプターの規格

アダプターの規格は JMS 1.0.2 規格に記載されています。その他のバージョンの規格については、サポートは検証されていませんが、現在、サポートを妨げる既知の問題はありません。

アダプターは、JMS 規格で定義された point-to-point (PTP) メッセージングおよびパブリッシュ/サブスクライブ (Pub/Sub) メッセージングの両方のインターフェースをサポートします。これらのスタイルは一般に、それぞれ、キュー・ベース・メッセージングおよびトピック・ベース・メッセージングと呼ばれています。アダプターの 1 つのインスタンスは、同時に 1 つのメッセージング・スタイルしかサポートしません (つまり、トピックとキューを混合して構成することはできません)。ただし、PTP または Pub/Sub のいずれかに構成されたインスタンスを使用し、アダプターの複数のインスタンスを平行して実行することにより、両方のメッセージング・スタイルをサポートすることはできます。

アダプターのプラットフォーム

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。

- Microsoft Windows 2000
- Microsoft Windows 2003
- IBM OS/400 V5R2、V5R3
- Red Hat Enterprise Linux AS 3.0
- SuSE Linux Enterprise Server 8.1 (SP3 を適用)

アダプターの依存関係

アダプターは、データベースを使用しません。また、データベースに依存しません。JMS プロバイダーおよび JNDI プロバイダーが必要とするすべてのクライアント・ライブラリーは、アダプター・クラスパスに入れる必要があります。これらのライブラリーはプロバイダーによって異なります。

ロケール依存データ

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

Java 仮想マシン (JVM) 内での Java ランタイム環境は、Unicode 文字コード・セットでデータを表します。Unicode には、最も広く知られている文字コード・セット (単一バイトおよびマルチバイトの両方) の文字のエンコードが含まれています。WebSphere Business Integration システムのほとんどのコンポーネントは Java で記述されています。したがって、ほとんどの統合コンポーネントの間でデータが転送されても、文字変換の必要はありません。

エラー・メッセージや通知メッセージを個々の国や地域に合った適切な言語で記録するには、個々の環境に合わせて `Locale` 標準構成プロパティーを構成する必要があります。構成プロパティーの詳細については、53 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

Adapter for JMS の用語

- **JMS プロバイダー**。JMS をインプリメントするメッセージング・システム。
- **メッセージ**。エンタープライズ・アプリケーションによって使用されるビジネス・データが入った要求およびイベント。
- **PTP**。point-to-point スタイルまたはキュー・ベースのメッセージング。
- **Pub/Sub**。パブリッシュ/サブスクライブ・スタイルまたはトピック・ベースのメッセージング。
- **JMS 宛先**。メッセージのソースまたはターゲットを表す。 PTP メッセージングでは、宛先はキューです。Pub/Sub では、宛先はトピックです。この用語は、特定の状況でキューまたはトピックが使用されるときに、説明および実際のプロパティー名を明記する両方の仕様で、広く使用されます。
- **ASI**。アプリケーション固有情報。ビジネス・オブジェクトおよびメタオブジェクトにおいて、セミコロンで区切られた name=value ペアとして表されるメタデータ。

Connector for JMS のアーキテクチャー

メッセージは、このアダプターとの関係においては、エンタープライズ・アプリケーションによって使用されるビジネス・データが入った要求およびイベントです。Message Oriented Middleware プロダクト (MOM) を使用すると、エンタープライズ・アプリケーションは、お互いに非同期方式でメッセージの送受信を行うことができます。 Java Message Service (JMS) API は、Java プログラムがこのようなメッセージング・システムと通信する方法を標準化するために定められました。以前は、しばしば、単一の特定 MOM システムと一緒に動作するメッセージング・クライアントが作成されました。アダプターなどの JMS クライアントは、一般に、JMS サポートを提供するすべてのメッセージング・システムを利用できます。 Adapter for JMS を使用すると、JMS 規格をサポートする、急増しているエンタープライズ・メッセージング・システムと統合できるようになります。

メッセージ処理

アダプターは、以下の 2 つの主要な操作をサポートします。

1. JMS 宛先からのメッセージの検索
2. JMS 宛先へのメッセージの配信

アダプターは、JMS プロバイダーへの接続を確立して両方の操作を行い、次に JMS API を使用して以下のことを行います。

- JMS 宛先の既存メッセージのポーリングおよび検索
- ブローカーが要求した新しいメッセージの生成および配信

これらの 2 つの操作については、4 ページの『イベント・メッセージの処理』 および 10 ページの『要求メッセージの処理』 で詳しく説明します。

イベント・メッセージの処理

コネクターは、新しいメッセージが、1 つ以上の JMS 宛先に配信されているか定期的に検査します。各ポーリング・サイクルで、コネクターは以下のことを行います。

1. JMS API を使用して、待機メッセージを検索する。
2. 構成済みデータ・ハンドラーを呼び出し、メッセージの内容をビジネス・オブジェクトに変換する。
3. サブスクライブしているビジネス・プロセスが処理できるように、イベント・ビジネス・オブジェクトを構成済み統合ブローカーに配信または公表する。

これらのステップについては、図 1 に示されています。また、以下で詳しく説明されています。

- 『イベント検出』
- 5 ページの『イベント状況およびリカバリー』
- 7 ページの『イベント検索』

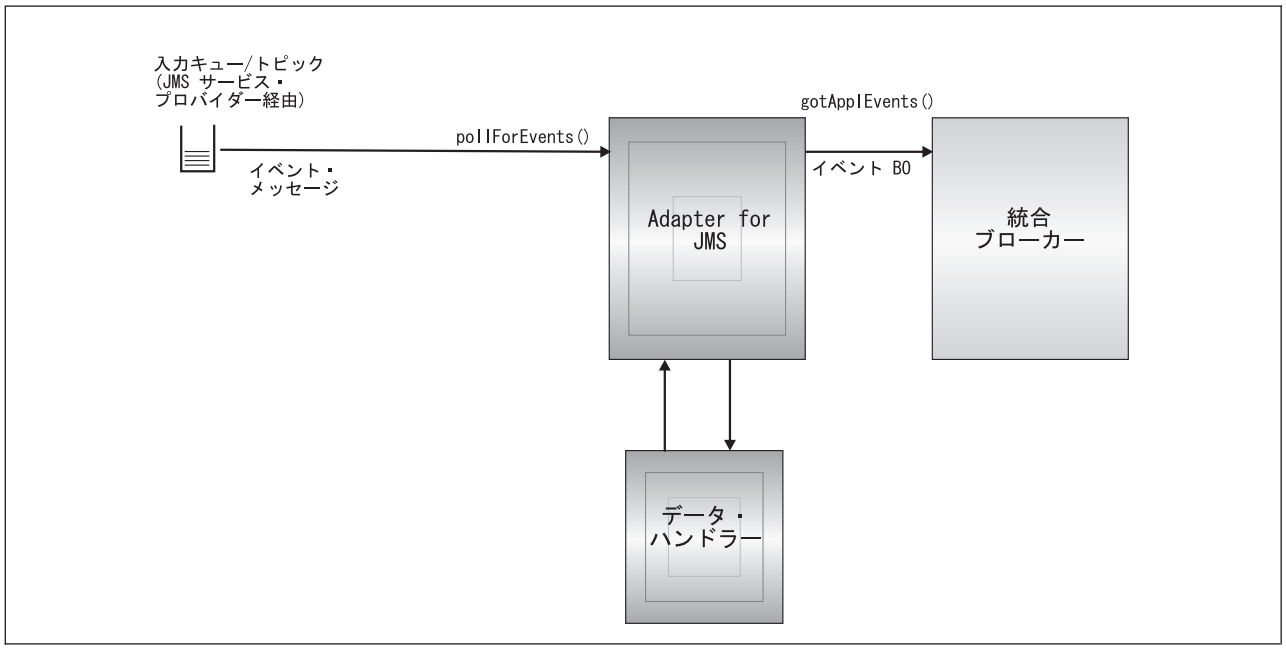


図 1. イベント・メッセージ・フロー

イベント検出

各イベント・ポーリング・サイクル中に、コネクターは、コネクター・プロパティー InputDestination によって指定された宛先で、メッセージの非ブロッキング読み取りを行います (コネクター・プロパティーの詳細については、21 ページの『コネクター・プロパティーの構成』を参照)。コネクターはメッセージを検索してから、ブローカーに公表します。

コネクターは `pollForEvents()` メソッドを使用して、一定の間隔でポーリングしてメッセージの有無を確認します。ポーリング・サイクルごとのメッセージ検索数

は、コネクター・プロパティー PollQuantity に指定されている最大数に制限されます。指定最大数に達する前に、すべての使用可能なメッセージを検索すると、コネクターは、それ以上のメッセージを待たずに、ポーリング・サイクルから即時に戻ります。

コネクター・プロパティー InputDestination で複数の宛先が指定されている場合、コネクターは、指定された各宛先をラウンドロビン方式でポーリングします。各宛先で *PollQuanity* に指定してある最大数のメッセージを検索し、ブローカーに公表します。PollQuantity に指定されている最大数に達する前にすべての宛先が空になると、コネクターはポーリング・サイクルから即時に戻ります。

例えば、次のようなシナリオでは、

- コネクターが、PollQuanity 値を 2、および入力キュー A、B、および C で構成されている
- キュー A にはメッセージが 2 つ、キュー B にはメッセージが 1 つ、キュー C にはメッセージが 5 つ含まれる

アダプターは以下のメッセージを検索します。

1 回のポーリング・サイクルにおいて:

1. キュー A の次のメッセージ (メッセージが 1 つ残る)
2. キュー B の次のメッセージ (空になる)
3. キュー C の次のメッセージ (メッセージが 4 つ残る)
4. キュー A の次のメッセージ (空になる)
5. コネクターがキュー B をチェックするが、空のまま
6. キュー C の次のメッセージ (メッセージが 3 つ残る)

各キューから最大数 (PollQuanity で指定) の 2 つのメッセージをポーリングしたので、アダプターはポーリング・サイクルから戻ります。

イベント状況およびリカバリー

イベント・メッセージの検索はトランザクションの一部です。トランザクションをコミットする前にコネクターが予期せずに終了する場合、トランザクションはロールバックされ、元のメッセージが復元されます。コネクター・フレームワークは現在、分散トランザクションをサポートしていないため、コネクターは、ブローカーにイベントを公表しても、ブローカーからの確認通知を受信する前に、予期せずに終了するか通信を切断する場合があります。この場合、イベントがブローカーに受信されたかどうかは、コネクターが入手できる情報からは確認できません。イベント・メッセージを失わないようにするために、ブローカーからイベントの受取を確認する応答をら受信するまでは、コネクターはトランザクションをコミットしません。コネクターがイベントを公表し、確認通知を受信するまでの間に障害が発生すると、トランザクションは自動的にロールバックされ、元のメッセージが復元されます。メッセージがブローカーによって処理されたかどうかがわからないため、そのようなイベントは未確定イベントと呼ばれます。

再始動時に、コネクターは、入力宛先からのメッセージの処理を開始し、未確定イベントを再サブミットします。この方式を使用すると、イベントを失うリスクはなくなりますが、同じイベントが 2 回公表される可能性は避けられません。

重複イベント配信のリスクを軽減したり、なくしたりする方法には、次の 2 つの方法があります。進行中の宛先の使用 (『進行中の宛先によるリカバリー』を参照) または保証付きイベント・デリバリーの使用 (『保証付きイベント・デリバリーによるリカバリー』を参照) です。

進行中の宛先によるリカバリー: 未確定イベントの処理を制御するには、コネクター・プロパティー InProgressDestination を指定して、別個の一時宛先を作成します。

注: 進行中の宛先のリカバリーは、Pub/Sub スタイルのメッセージングではサポートされていません。

ブローカーにイベントを公表する前に、コネクターは、イベント・メッセージを入力宛先から進行中の宛先へ移動します。ブローカーから確認通知を受信したら、コネクターは、進行中の宛先からメッセージを削除します。これによって、処理されていない未確定メッセージを分離できます。始動時に進行中の宛先にメッセージがあった場合、コネクターは、これらのメッセージは、予期せずに終了したコネクターの以前のインスタンスから残されたものであるとみなすことができるため、安全です。そのようなメッセージに、コネクターが別の処置を行うように指定できます (重複イベント通知が受諾不能な場合)。これを行うには、以下の 4 つのオプションのいずれかを、コネクター構成プロパティー InDoubtEvents に指定します。

- **Fail on startup** 初期設定中に進行中の宛先からメッセージが検出された場合に、コネクターはエラー・ログを記録し、すぐ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、メッセージを調べ、これらのメッセージをすべて削除するか、それらを別の場所に移動するか、いずれかの適切な処置を行います。
- **Reprocess** 初期設定中に進行中の宛先からメッセージが検出された場合に、コネクターは、以降のポーリングでこれらのメッセージを最初に処理します。進行中の宛先のすべてのメッセージの処理が完了すると、コネクターは入力宛先のメッセージの処理を開始します。
- **Ignore** 初期設定中に進行中の宛先からメッセージが検出された場合に、コネクターはそれらを無視しますが、シャットダウンはしません。
- **Log error** Log error オプションを使用すると、初期設定中に進行中の宛先からメッセージが検出された場合に、コネクターはエラー・ログを記録しますが、シャットダウンはしません。

詳細については、 24 ページの『InDoubtEvents』を参照してください。

保証付きイベント・デリバリーによるリカバリー: 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、イベントが逸失したり、イベントが 2 度送信されたりするのを防ぐことができます。コネクター・フレームワークは、コンテナー管理イベント (CME) および重複イベント除去 (DEE) の、2 つの機構によって保証付きイベント・デリバリーをサポートします。

コンテナー管理イベント (CME): コネクターが PTP スタイル・メッセージングに構成されている場合に、CME を使用できます。Pub/Sub スタイル・メッセージング用に構成されている場合、コネクターは CME をサポートしません。

ContainerManagedEvents コネクター・プロパティーの詳細については、 59 ページの『ContainerManagedEvents』を参照してください。

重複イベント除去 (DEE): JMS アダプターに保証付きイベント・デリバリーをインプリメントするときは、DEE 方式をお勧めします。DEE は Pub/Sub スタイルのメッセージングをサポートする唯一の方法でもあります。

DEE では、コネクターは、ブローカーに公表する各イベントに固有 ID を組み込みます。フレームワークは、コネクターが同じイベント ID を連続してサブミットしていないかチェックします。連続してサブミットされると、フレームワークは、コネクターが同じイベントを 2 度公表しているとみなし、2 番目のサブミットを廃棄します。PTP スタイルのメッセージングに関しては、DEE は進行中の宛先から、またはその宛先へのメッセージをコピーするオーバーヘッドをかなり削減します。

このコネクターは、ビジネス・オブジェクトをブローカーに公表するときに、すべてのイベントのメッセージ ID を組み込みます。通信障害または予期せぬ終了によって、コネクターがイベントをブローカーに正常に送付できない場合は、前に説明したように、元のメッセージが入力キューにロールバックされます。コネクターは再始動時に、すべての未確定メッセージを含め、キューのイベントの再サブミットを開始します。 DEE が使用可能になっていれば、以前ブローカーに正常に到達した未確定メッセージは、すべて廃棄されます。これによって、各メッセージをブローカーに 1 度だけ送付するようにできます。

DEE を使用する場合、コネクターがオフラインの間は、宛先内のメッセージの順序を変更しないようにする必要があります。DEE は、アダプターによって検索された最後のメッセージ ID のみを記録します。アダプターが再始動する前に、高い優先順位を持った新しいメッセージが、最後の未確定メッセージの順序をキュー内で押し下げるなどの場合に、DEE は失敗します。

DuplicateEventElimination コネクター・プロパティーの詳細については、61 ページの『DuplicateEventElimination』を参照してください。

イベント検索

イベント検索には、コネクターによるイベントの典型的処理が含まれます。イベント検索は、着信イベントが検出されると始まり、それがターゲット・アプリケーションに適したフォーマットに変換され、指定された統合ブローカーに正常に配信されると終了します。コネクターは、すべてのイベントをブローカーに非同期的に (「応答不要送信」で) 配信します。

以下のセクションでイベント検索について説明します。

- 『メタデータおよびメタオブジェクト』
- 8 ページの『ビジネス・オブジェクトのマッピング』
- 9 ページの『メッセージ・ヘッダー・マッピングの理解』
- 10 ページの『アーカイブ』
- 10 ページの『エラー・リカバリー』

メタデータおよびメタオブジェクト: メッセージを正常にビジネス・オブジェクトに変換したり、その逆を行ったりするには、コネクターは、メタデータと呼ばれる追加情報を必要とします。メタデータは、オブジェクト、メッセージ、またはアプリケーション内のデータをどのように表現するか、またはそれらをどのように処理するかを説明します。メタデータには、コネクターが宛先 XYZ からメッセージを検索した場合、どのビジネス・オブジェクトを作成するか、または動詞 `Create` を

持ったタイプ Customer の要求ビジネス・オブジェクトをシリアライズするときに、どのデータ・ハンドラーを使用するか、などの詳細が含くまれています。

属性、プロパティー、動詞、およびアプリケーション固有情報が、ビジネス・オブジェクト定義のメタデータを構成します。さらに、宛先、データ・フォーマット、データ・ハンドラーなどのメタデータを含んだ、1 つ以上のメタオブジェクトを指定できます。

メタオブジェクトには、静的タイプおよび動的タイプの 2 つのタイプがあります。インプリメンテーションのときには、静的メタオブジェクトを作成します。静的メタオブジェクトには、コネクターがサポートしなければならない各ビジネス・オブジェクト・タイプにメタデータを提供する属性が含まれます。静的メタオブジェクトはコネクター固有プロパティーで指定され、初期設定のときにコネクターによって読み取られます。メタオブジェクト・プロパティーの概要、およびそれらがメッセージ変換にどのような影響を与えるかについては、『ビジネス・オブジェクトのマッピング』および 9 ページの『メッセージ・ヘッダー・マッピングの理解』を参照してください。

もう一つのメタオブジェクト・タイプは動的メタオブジェクトです。このメタオブジェクトを使用すると、要求処理のときに、要求ごとに、アダプターが使用するメタデータを変更してビジネス・オブジェクトを処理できます。イベント処理のときに、動的メタオブジェクトは、イベントに関するトランスポート固有情報 (メッセージ ID、優先順位など) を保持するコンテナーとして機能します。このため、ダウンストリーム・ビジネス・プロセスはそれらのビジネス・ロジックでその情報を使用できます。動的メタオブジェクトは、イベント (または要求) のトップレベル・オブジェクトで定義される、特別にマークされた子オブジェクトとして表されます。

メタオブジェクトは、同じインプリメンテーションにおいて、どちらか 1 つを使用することも、両方を使用することもできます。一般に、動的メタオブジェクトで指定された値は、静的メタオブジェクトで指定された値に優先します。静的および動的メタオブジェクトの構成については、28 ページの『メタオブジェクトの構成』を参照してください。

ビジネス・オブジェクトのマッピング: メッセージの検索のときに、コネクターは、メッセージをどのビジネス・オブジェクトにマップすべきかを確認しようとします。

デフォルトでは、コネクターは、コネクター・プロパティーに設定されているデータ・ハンドラーが、ビジネス・オブジェクト・タイプを決定できるようにします。コネクターは、メッセージ本文をデータ・ハンドラーに渡し、データ・ハンドラーが戻したビジネス・オブジェクトをブローカーに公表します。データ・ハンドラーが適切なビジネス・オブジェクトを決定できない場合、コネクターはイベントに失敗します。

コネクター構成プロパティー ConfigurationMetaObject に静的メタオブジェクトが指定されている場合、コネクターはこのオブジェクトを検索して、入力フォーマットまたは入力宛先に関してメッセージに一致するルールを見付けます。メタオブジェクトで指定されたルールが入力フォーマットおよび入力宛先の両方を指定している場合、メッセージがそれらのプロパティーの両方に一致する場合にのみ、コネクタ

ーはこのルールに従います。これらのプロパティーのどちらかが欠落している場合、コネクターは指定されたプロパティーのみを使用します。

例えば、入力フォーマット Cust_In を持った、入力宛先 MyInputDest のメッセージは、以下の静的メタオブジェクト・ルールに一致します。

1. InputFormat=Cust_In;InputDestination=MyInputDest
2. InputFormat=Cust_In
3. InputDestination=MyInputDest

イベント・メッセージを 1 つのルールに一致させることができる場合、コネクターは、ビジネス・オブジェクトの新しいインスタンスを作成し、それをメッセージ本文と一緒に、ルールで指定されたデータ・ハンドラーに渡すことによって、このビジネス・オブジェクトを書き取らせます。データ・ハンドラーがルールに指定されていない場合、コネクターは、コネクター構成プロパティーで指定されたデフォルトのデータ・ハンドラーを使用します。

アダプターが、イベント・メッセージを複数のルールに一致させることができる場合、または 1 つのルールにも一致させることができない場合、コネクターは、コネクター構成プロパティーで指定されたデータ・ハンドラーにメッセージ本文のみを渡すことによって、データ・ハンドラーが、ビジネス・オブジェクト・タイプを決定できるようにします。

メッセージ・ヘッダー・マッピングの理解: イベント・メッセージをビジネス・オブジェクトに変換するために、コネクターは、ビジネス・オブジェクトに関するメタデータとメッセージに関するメタデータを比較し、それらをマッピングします。7 ページの『メタデータおよびメタオブジェクト』で説明したように、ビジネス・オブジェクトに関するメタデータは、ビジネス・オブジェクト定義 (アプリケーション固有情報および子動的メタオブジェクト)、コネクター・プロパティー、および静的メタオブジェクトの中にあります。メッセージ・メタデータはメッセージ・ヘッダーの中にあります。

トランスポート固有メッセージ・ヘッダー情報へアクセスしたり、メッセージ・トランスポートの詳細情報を入手したり、それを詳細に制御したりするには、ビジネス・オブジェクト定義の子である動的メタオブジェクトに属性を追加します。属性を追加すると、メッセージ・ヘッダーから読み取りができるようになり、オプションで書き込みもできるようになるため、メッセージ・メタデータを変更できるようになります。そのような変更では、JMS プロパティーを変更したり、要求ごとに宛先を制御したり (アダプター・プロパティーで指定されたデフォルトの宛先を使用せずに)、メッセージの CorrelationID を再ターゲットしたりできます。ビジネス・オブジェクト定義の子である動的メタオブジェクトにそのようなプロパティーを指定すると、コネクターは、メッセージ・ヘッダーでそれらに対応するものをチェックし、メッセージ・ヘッダーの内容に基づいて、動的メタオブジェクトにデータを取り込みます。 1 つまたはすべてのサポートされた動的メタオブジェクト属性を定義できます。コネクターはそれに従って、メタオブジェクトにデータを取り込みます。読み取りまたは書き込みができるメッセージ・ヘッダー・プロパティーのリストを含め、詳細については、36 ページの『ポーリング中の動的な子メタオブジェクトの含まれるデータ』を参照してください。

アーカイブ: コネクター固有プロパティー ArchiveDestination を指定すると、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この宛先に置きます。ArchiveDestination が未定義の場合、正常に処理されたメッセージは廃棄されます。詳細については、 21 ページの『コネクター固有プロパティーの構成』を参照してください。

エラー・リカバリー: 入力宛先からの読み取りでエラーを検出すると、コネクターは、定数値 APPRESPONSETIMEOUT をすぐにブローカーに戻します。これにより、コネクターは終了し、場合により再始動します。一般に、そのようなリカバリー不能エラーは、JMS プロバイダーへの接続が切断されたか、あるいはコネクターが認識できないか、認識してもリカバリー不能 (トランザクションの失敗など) とみなした、JMS プロバイダーによって報告された内部エラーかのいずれかが原因です。

インバウンド・メッセージをイベント・ビジネス・オブジェクトに変換しているときにエラーを検出した場合 (データ・ハンドラーが無効なメッセージ・フォーマットを報告する場合など)、コネクターはイベントに失敗し、理由を説明する該当するエラー・メッセージのログを記録します。コネクター・プロパティー ErrorDestination が定義されており、有効な場合、コネクターは、失敗したメッセージのコピーをこのエラー宛先に置きます。そうでない場合、メッセージは廃棄されます。

コネクターがイベント・ビジネス・オブジェクトを公表した後にブローカーがエラーを報告する場合、コネクターはイベントに失敗し、ブローカーによって報告されるエラー・メッセージのログを記録します。コネクター・プロパティー ErrorDestination が定義されており、有効な場合、コネクターは、失敗したメッセージのコピーをこの宛先に置きます。そうでない場合、メッセージは廃棄されます。

メッセージのビジネス・オブジェクトを決定できない場合、またはメッセージをブローカーに公表しても、ブローカーがそのメッセージはサポートされていないと報告する場合、コネクターは、メッセージはアンサブスクライブされているとみなします。コネクター・プロパティー UnsubscribedDestination が定義されており、有効な場合、コネクターは、アンサブスクライブされたメッセージのコピーをこの宛先に置きます。そうでない場合、メッセージは廃棄されます。

要求メッセージの処理

ビジネス・オブジェクト要求がコネクターに送信されると、コネクターは、ターゲット宛先で新しいメッセージを作成します。メッセージ・ヘッダーには、要求メタオブジェクトで指定されたユーザー定義の値とコネクター・プロパティーによって指定されたデフォルト・パラメーターを組み合わせたデータが読み込まれます。メッセージの本文には、構成されたデータ・ハンドラーを介して要求ビジネス・オブジェクトを渡すことによって生成された結果内容のデータが取り込まれます。

図 2 に、メッセージ要求通信を示します。`doVerbFor()` メソッドがブローカーからビジネス・オブジェクトを受信すると、コネクターはビジネス・オブジェクトをデータ・ハンドラーに渡します。データ・ハンドラーはビジネス・オブジェクトを適切なテキストに変換し、コネクターはそれをメッセージとして宛先に発行します。

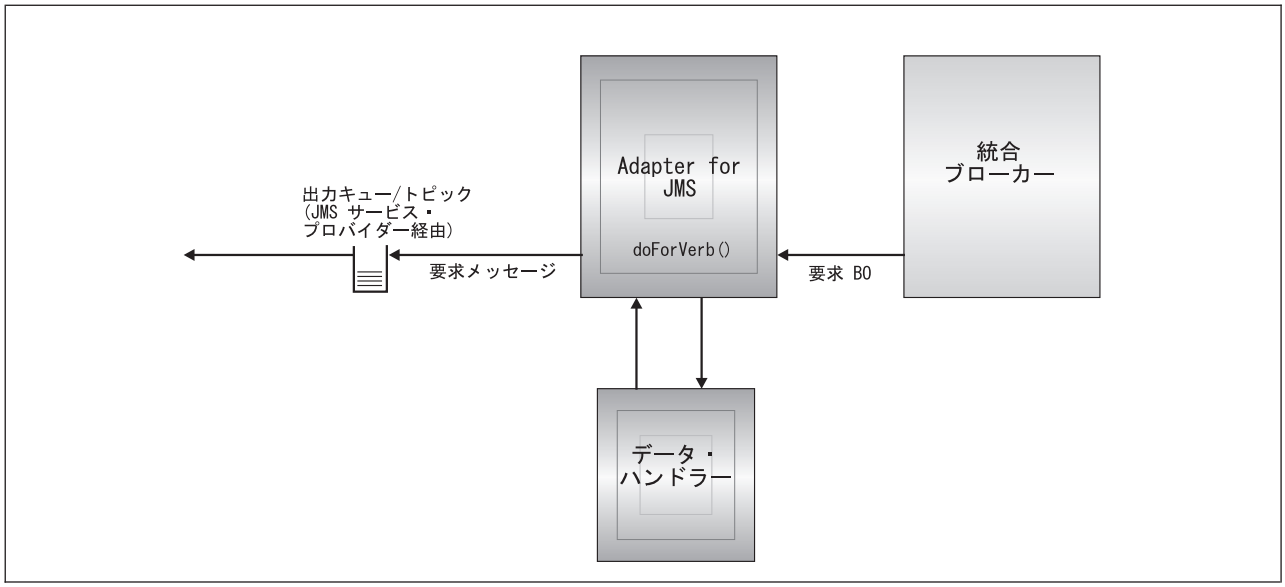


図 2. 要求フロー

要求の処理では、コネクターは 2 つのタイプの処置を行うことができます。 1 つ目は下記で非同期処理として説明するものであり、コネクターはメッセージをターゲット宛先に置いて、正常に戻ります。一般に、これは「応答不要送信」と呼ばれます。 2 つ目は下記で同期処理として説明するものであり、コネクターはメッセージを同じようにターゲット宛先に置きますが、ターゲット・アプリケーションによって応答が戻されるのも待ちます。

処理モードは数値プロパティー ResponseTimeout によって決まります。これは、ビジネス・オブジェクト要求の動的メタオブジェクトまたは静的メタオブジェクトのいずれかで指定されます。このプロパティーが定義されていないか、-1 に設定されている場合、コネクターは要求を非同期的に配信します。このプロパティーが 0 以上の場合、アダプターは、要求を同期的に処理し、ターゲット・アプリケーションが応答メッセージを戻すのをそのミリ秒以上待ちます。図 2 に示した要求処理を、以下で詳しく説明します。

- 『動詞サポート』
- 『非同期処理』
- 13 ページの『同期処理』

動詞サポート

コネクターは、要求ビジネス・オブジェクトで指定される動詞にセマンティック値は置きません。コネクターは同じ処置を行います。つまり、指定される動詞に関係なく、メッセージを JMS 宛先に置きます。

非同期処理

非同期処理では、コネクターは要求ビジネス・オブジェクトをメッセージに変換し、そのメッセージをターゲット宛先に置いてから、すぐブローカーに戻ります。要求が成功するか失敗するかは、すべて、メッセージを JMS 宛先に置くコネクターの能力に基づいています。この配信が成功しても、ターゲット・アプリケーションにメッセージがあるわけでもメッセージを受信するわけでもないことに注意してください。メッセージング・システムが非同期的性質を持つ場合、ターゲット・ア

プリケーションがメッセージを処理できるようになるまで、または有効期限切れになる (そのように構成されている場合) まで、メッセージは JMS プロバイダーに無期限にとどまる場合があります。

コネクターはまず、構成されたデータ・ハンドラーを使用して、要求ビジネス・オブジェクトをテキストにシリアライズします。コネクターは、以下に指定されたデータ・ハンドラーを使用します (優先順)。

1. 動的メタオブジェクト
2. 静的メタオブジェクト
3. コネクター構成プロパティー

コネクターは、シリアライズされたビジネス・オブジェクト・データの入った新しいメッセージをメッセージの本文として作成します。コネクターは、以下の表の説明に従って、メッセージ・ヘッダーにデータを読み込みます。動的メタオブジェクトまたは静的メタオブジェクトのどちらにおいてもプロパティーを指定できるすべての場合で、動的メタオブジェクトで指定された値は、静的メタオブジェクトで指定された値に優先します。メタオブジェクトで指定できるプロパティーの説明およびリストについては、28 ページの『メタオブジェクトの構成』を参照してください。

表 1. 非同期要求処理での JMS メッセージ・ヘッダーの読み込み

メタオブジェクト・プロパティー	プロパティーが未定義の場合のデフォルト処置	プロパティーが定義されている場合に行われる処置
OutputFormat	コネクターはメッセージ・フォーマットを指定しない	コネクターはメッセージ・フォーマットにこの値を指定する
CorrelationID	コネクターはメッセージ・ヘッダーでこの値をブランクにする	コネクターは、要求メッセージ・ヘッダーで相関 ID にこの値を指定する
ReplyToDestination	コネクターはメッセージ・ヘッダーでこの値をブランクにする	コネクターは、要求メッセージ・ヘッダーで応答宛先にこの値を指定する
Priority	コネクターは、JMS プロバイダーがデフォルト優先順位を使用できるようにする	コネクターは、この値を使用してメッセージ優先順位の数値を設定する
JMSProperties	なし	コネクターは、指定された JMS プロパティーをメッセージ・ヘッダーの JMS プロパティーにマップする

メタオブジェクトの次の属性が、メッセージの配信方法を決めます。

表 2. 宛先への非同期配信

メタオブジェクト・プロパティー	プロパティーが未定義の場合のデフォルト処置	プロパティーが定義されている場合に行われる処置
OutputDestination	値が必要	メッセージのターゲット宛先
DeliveryMode	コネクターは、JMS プロバイダーがメッセージ・パーシスタンスを書き取らせることができるようにする	コネクターは、ユーザーの指示に従って、メッセージを永続的または非永続的に書き込む

コネクターが、要求メッセージを出力 (ターゲット) 宛先に正常に配信できたかどうかによって、以下のいずれかのコードがブローカーに戻されます。

表 3. 非同期戻りコード

コネクターの処置	ブローカーへの戻りコード
メッセージを正常にターゲット宛先へ配信	SUCCEED
不適切または不完全なメタデータ、データ・ハンドラーの失敗、または一般的な処理問題などのリカバリー可能エラーにより、配信が失敗	FAIL
JMS プロバイダーによって報告される、接続の失敗などのリカバリー不能エラーにより、配信が失敗	APPRESPONSETIMEOUT

同期処理

同期処理では、コネクターは要求をターゲット宛先へ配信してから、2 番目の宛先で応答メッセージを待ちます。要求メッセージの作成は非同期処理の場合と同じです。ただし、コネクターは、メタオブジェクトの以下の追加属性もチェックします。

表 4. 同期メタオブジェクト・プロパティー

メタオブジェクト・プロパティー	プロパティーが未定義の場合のデフォルト処置	プロパティーが定義されている場合に行われる処置
ResponseTimeout	値が必要	アダプターが、応答メッセージが戻るのを待つ最小時間 (ミリ秒単位)
TimeoutFatal	ResponseTimeout で指定された時間までに応答を受信できない場合、コネクターは APPRESPONSETIMEOUT をブローカーに戻す。これにより、通常、コネクターは終了する。	応答を受信できない場合、コネクターは要求に失敗するが (FAIL をブローカーに戻す)、終了はしない。

ターゲット宛先へのメッセージの配信は、以下の点を除いて、非同期処理の場合と同じです。

表 5. 宛先への同期配信

メタオブジェクト・プロパティー	プロパティーが未定義の場合のデフォルト処置	プロパティーが定義されている場合に行われる処置
ReplyToDestination	非同期の場合と同じ	コネクターは、要求メッセージのこのフィールドに、コネクター固有プロパティー ReplyToDestination の値を読み込む

コネクターは、メタオブジェクト属性 ResponseTimeout で指定された時間以上、ReplyToDestination で指定されたターゲット・アプリケーションからの応答メッセージを待ちます。応答がその時間内に戻らない場合、コネクターはタイムアウトになり、エラーを報告します。

応答基準: コネクターは、応答宛先の最初のメッセージが正しい応答メッセージであるとはみなしません。その代わり、JMS 要求応答規則に従って、要求のメッセージ ID に一致する相関 ID を持つ最初のメッセージを探します。つまり、要求メッ

セージを受信するアプリケーションは、要求メッセージ ID に等しい相関 ID を持つ応答メッセージを作成し、そのメッセージを、要求メッセージによって指定された応答宛先に置く必要があります。

すべてのアプリケーションが相関 ID を使用する規則に従って、要求メッセージと応答メッセージをマップするわけではありません。その場合、コネクターは、応答メッセージを識別するカスタム基準を受け入れます。

コネクターは、同期要求処理の対象となるビジネス・オブジェクトを受信するときに、動詞のアプリケーション固有情報に、名前と値のペア response_selector= が存在するかどうかを検査します。そのような名前と値のペアが存在しない場合、コネクターは、既に説明したように、メッセージ相関 ID を使用して応答メッセージを識別します。

応答セレクターの名と値のペアが定義されている場合、コネクターは、応答メッセージを識別できる、JMS メッセージ・セレクター・ストリングを表す値とみなします。以下にいくつかの使用例を示します。JMS メッセージ・セレクター構文の詳細については、JMS API 仕様を参照してください。JMS メッセージ・セレクター構文はコネクターによって解析されないことに注意してください。その代わり、構文は JMS プロバイダーによって解釈されます。コネクターは、JMS プロバイダーが、メッセージをフィルタリングする手段としてセレクターを使用できるようにします (データベースのクエリーに類似)。

例えば、名前と値のペアが入った、次の動詞アプリケーション固有情報は、

```
response_selector=JMSType = 'xmlResponse'
```

応答メッセージが、セレクター・ストリング JMSType = 'xmlResponse' に一致しなければならないことをコネクターに知らせます。コネクターはこのセレクターを JMS プロバイダーに提供します。次に JMS プロバイダーは、配信された最初のメッセージを、メッセージの JMS タイプ・フィールドが xmlResponse に等しい応答宛先に戻します。

すべての場合で、メッセージ・セレクター・ストリングは、1 つだけの応答を一意に識別できなければなりません。応答セレクターの基準を満たした応答宛先に複数のメッセージが配信されると、アダプターは最初のメッセージのみを検索します。基準に一致する可能性のあるその他の応答メッセージは無視されます。

実行時にメッセージ・セレクターが固有なものになるように、コネクターは、メッセージ・セレクター自身への属性値の動的置換をサポートしています。これを行うには、応答セレクターで、整数を中括弧で囲んだ形式 (″{1}″) のプレースホルダーを指定する必要があります。この後にコロンを置き、置換に使用する属性をコンマで区切ってリストしてください。プレースホルダーの整数は、置換に使用する属性に対する指標として機能します。

例えば、以下のメッセージ・セレクターでは、

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

コネクターは、トークン {1} を、子オブジェクト MyDynamicMO の属性 CorrelationID の値で置き換えることを知らせます。属性 CorrelationID が 123ABC の値を持つ場合、コネクターは、次のメッセージ・セレクターを生成し、それを使用します。

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

下に示すように、複数の置換を指定することもできます。

```
response_selector=Name LIKE '{1}'AND Zip LIKE '{2}':PrimaryID,Address[4].AddressID
```

この例では、コネクターは、'{1}' を、トップレベル・ビジネス・オブジェクトの属性 PrimaryID の値で置き換え、'{2}' を、子コンテナー・オブジェクト Address の 5 番目 (ベース 0) にある AddressID の値で置き換えます。この方法では、応答メッセージ・セレクターでビジネス・オブジェクトおよびメタオブジェクトの任意の属性を参照できます。

メッセージ・セレクターでリテラル値「{」を指定するには、その代わりに「{{」を使用します。例えば、以下のセレクターでは、

```
response_selector=PrimaryID LIKE {{1}
```

アダプターは、これを以下のリテラル値として認識します。

```
PrimaryID LIKE {1}
```

この場合、コネクターは、値 '{1}' での置換は行いません。

コネクターは、属性値で「{」、「}」、「:」、「;」などの特殊文字を検出した場合は、それらの文字を照会ストリングに直接挿入します。これにより、アプリケーション固有情報の区切り文字としても機能する特殊文字を照会ストリングに含めることができます。例えば、以下のセレクターでは、

```
Response_selector=PrimaryID = '{1}':Foo
```

属性 Foo が {A:B};{C:D} の値を持つ場合、以下のようなリテラル・メッセージ・セレクターに変換されます。

```
PrimaryID = '{A:B};{C:D}'
```

応答処理: 応答メッセージを受け取ったときに行う処置を確認するために、コネクターは、コネクター・プロパティー MessageResponseResultProperty によって指定された JMS 結果プロパティーを検査します。この JMS プロパティーの値によって、コネクターは、メッセージ本文にビジネス・オブジェクトとエラー・メッセージのどちらが含まれているかを予想します (下の表を参照)。すべての場合に、コネクターは対応する戻りコードをブローカーに戻します。メッセージで JMS 結果プロパティーが VALCHANGE に等しい場合、コネクターは、この表で説明された VALCHANGE についての処置を行い、ブローカー定数 VALCHANGE に対応した数値をブローカーに戻します。

表 6. 応答メッセージの処理

JMS 結果プロパティーの値	コネクターの処置
SUCCESS	要求ビジネス・オブジェクトを変更せず、単に、正常にブローカーに戻ります。

表 6. 応答メッセージの処理 (続き)

JMS 結果プロパティーの値	コネクターの処置
VALCHANGE MULTIPLE_HITS	要求ビジネス・オブジェクトに応答メッセージ本文の内容を再び読み込みます。応答メッセージ本文が空の場合、要求ビジネス・オブジェクトは変更されません。 要求ビジネス・オブジェクトの動的メタオブジェクトに、応答メッセージの JMS ヘッダー・フィールドを再び読み込みます。
FAIL FAIL_RETRIEVE_BY_CONTENT BO_DOES_NOT_EXIST UNABLE_TO_LOGIN VALDUPES	応答にデータが読み込まれている場合、コネクターは応答をエラー・メッセージとみなし、それをブローカーに戻します。応答メッセージ本文が空の場合、コネクターは一般エラー・メッセージをブローカーに戻します。
APPRESPONSETIMEOUT	APPRESPONSETIMEOUT がブローカーに戻ると、通常、アダプター・エージェントが終了する点を除いて、上記と同じです。
未定義または認識できない値	コネクターは要求に失敗します。

エラー処理: ターゲット宛先との間での要求メッセージの読み取りまたは書き込みのとき、または応答メッセージの検査のときに (該当する場合) エラーを検出すると、コネクターは、APPRESPONSETIMEOUT をすぐにブローカーに戻します。これにより、アダプターは終了するか、場合により再始動します。一般に、そのようなリカバリー不能エラーは、JMS プロバイダーへの接続が切断されたか、あるいはコネクターが認識できないか、認識してもリカバリー不能 (トランザクションの失敗など) とみなした、JMS プロバイダーによって報告された内部エラーかのいずれかが原因です。

ビジネス・オブジェクトをメッセージに変換しているとき、またはその逆の変換のときにエラーを検出した場合 (データ・ハンドラーが無効なメッセージ・フォーマットを報告する場合など)、コネクターは要求に失敗し、理由を説明した該当するエラー・メッセージのログを記録します。

イベント失敗のシナリオを含む詳細については、 49 ページの『エラー処理』を参照してください。

第 2 章 アダプターのインストールおよび構成

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

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 JMS は、以下のアダプター・フレームワークと統合ブローカーでサポートされています。

- アダプター・フレームワーク: WebSphere Business Integration Adapter Framework バージョン 4.3.1
- 統合ブローカー: InterChange Server Express バージョン 4.3.1

例外については、「リリース情報」を参照してください。

前提条件

前提条件ソフトウェア

- コネクターは JMS 1.02 をサポートしています。
- また、以下のコンポーネントが必要です。
  - Java CDK (「*WebSphere Business Integration Server Express インストール・ガイド*」参照)
  - JMS および JNDI ライブラリー
- コネクターは、以下のプラットフォームで実行できます。
  - Microsoft Windows 2000

– Microsoft Windows 2003
– IBM OS/400 V5R2、V5R3
– Red Hat Enterprise Linux AS 3.0
– SuSE Linux Enterprise Server 8.1 (SP3 を適用)

インストール・タスク

Adapter for JMS をインストールするには、以下の作業を実行する必要があります。

- **統合ブローカーのインストール** この作業では、WebSphere Business Integration Server Express のインストールと始動を行います。作業の詳細については、「*WebSphere Business Integration Server Express インストール・ガイド*」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。『アダプターと関連ファイルのインストール』を参照してください。

アダプターと関連ファイルのインストール

アダプターのインストール方法の詳細については、以下の WebSphere Business Integration Server Express Infocenter のサイトにある Windows 版、Linux 版、または OS/400 版の「WebSphere Business Integration Server Express インストール・ガイド」を参照してください。

http://www.ibm.com/websphere/wbiserverexpress/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、Windows、OS/400、および Linux プラットフォーム上でのアダプターのインストール済みファイルの構造について説明します。

コネクターが使用する Windows のファイル構造を 19 ページの表 7 に、Linux のファイル構造を 19 ページの表 8 に、OS/400 のファイル構造を 20 ページの表 9 に示します。

コネクター・コンポーネントのインストール方法については、以下のガイドを参照してください。

- *クイック・スタート・ガイド*
- *IBM WebSphere Business Integration Server Express and Express Plus インストール・ガイド*

Windows のコネクター・ファイル構造

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

このユーティリティーは、コネクター・エージェントを *ProductDir*¥connectors¥JMS ディレクトリーにインストールして、「スタート」メニューにコネクター・エージェントへのショートカットを追加します。*ProductDir* は、アダプター製品がインストールされているディレクトリーを表していることに

注意してください。環境変数には、*ProductDir* ディレクトリーへのパス (デフォルトでは IBM¥WebSphereServer) が含まれています。

表 7 に、コネクターが使用する Windows ファイル構造が記載されており、インストーラーを介したコネクターのインストールを選択した際に自動的にインストールされるファイルを示します。

表 7. コネクター用としてインストールされた Windows ファイル構造

***ProductDir* のサブディレクトリー**	**説明**
connectors¥JMS¥CWJMS.jar	JMS コネクターに使用されるクラスを含みます。
connectors¥JMS¥jms.jar	コネクターが必要とするサード・パーティーのライブラリー。
connectors¥JMS¥start_JMS.bat	コネクター (Windows 2000/2003) の始動スクリプト。
connectors¥JMS¥start_JMS_service.bat	サービスとしてコネクターの始動スクリプト。
connectors¥messages¥JMSConnector.txt	コネクターのメッセージ・ファイル。
repository¥JMS¥CN_JMS.txt	コネクターのリポジトリー定義。
connectors¥JMS¥Samples¥WebSphereICS¥JMSSample.jar	サンプル成果物を収容する JAR ファイル

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。

Windows の場合、インストーラーによって、**「プログラム」>「IBM WebSphere Business Integration Express」>「アダプター」>「コネクター」**を選択することにより表示可能なコネクター・ファイルのアイコンが追加されます。

Linux コネクターのファイル構造

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

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

表 8 には、コネクターが使用する Linux ファイルの構造が説明されており、インストーラーによるコネクターのインストールを選択したときに自動的にインストールされるファイルが示されています。

表 8. コネクター用としてインストールされた Linux ファイル構造

***ProductDir* のサブディレクトリー**	**説明**
connectors/JMS/CWJMS.jar	JMS コネクターに使用されるクラスを含みます。
connectors/JMS/jms.jar	コネクターが必要とするサード・パーティーのライブラリー。
connectors/JMS/start_JMS.sh	コネクターのシステム始動スクリプト。
connectors/messages/JMSConnector.txt	コネクターのメッセージ・ファイル。

表 8. コネクター用としてインストールされた *Linux* ファイル構造 (続き)

***ProductDir* のサブディレクトリー**	**説明**
repository/JMS/CN_JMS.txt	コネクターのリポジトリー定義。
connectors/JMS/Samples/WebSphereICS/JMSConnector.cfg	コネクター構成ファイルのサンプル

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。

Linux の場合は、「connector_manager」コマンドによってコネクターを始動する必要があります。command.

OS/400 コネクターのファイル構造

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

このユーティリティーは、コネクター・エージェントを */QIBM/ProdData/WBIServer43/product* ディレクトリーにインストールされます。

表 9 に、コネクターが使用する OS/400 ファイル構造が記載されており、インストーラーを介したコネクターのインストールを選択した際に自動的にインストールされるファイルを示します。

表 9. コネクター用としてインストールされた *OS/400* ファイルのファイル構造

***ProductDir* のサブディレクトリー**	**説明**
connectors/JMS/CWJMS.jar	JMS コネクターに使用されるクラスを含みます。
connectors/JMS/jms.jar	コネクターが必要とするサード・パーティーのライブラリー。
connectors/JMS/start_JMS.sh	コネクターのシステム始動スクリプト。
connectors/messages/JMSConnector.txt	コネクターのメッセージ・ファイル。
repository/JMS/CN_JMS.txt	コネクターのリポジトリー定義。
connectors/JMS/Samples/WebSphereICS/JMSConnector.cfg	コネクター構成ファイルのサンプル

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。

OS/400 の場合にコネクターをすばやく始動するには、WebSphere Business Integration Console 機能を使用します。コンソールの詳細については、コンソールに付属のオンライン・ヘルプを参照してください。

コネクター構成

インストールしたアダプターは、コネクターを構成する必要があります。構成するには、以下のセクションで説明されている作業を行う必要があります。

- 21 ページの『コネクター・プロパティーの構成』
- 27 ページの『メッセージ・スタイルの構成』
- 27 ページの『JNDI の構成』

- 28 ページの『メタオブジェクトの構成』
- 39 ページの『開始スクリプトの構成』

コネクター・プロパティーの構成

コネクターには、以下のセクションで説明されている 2 つのタイプの構成プロパティーがあります。

- 『標準コネクター・プロパティーの構成』
- 『コネクター固有プロパティーの構成』

アダプターを実行する前に、これらのプロパティーの値を設定する必要があります。

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

- Connector Configurator Express の説明と段階的な手順については、69 ページの『付録 B. Connector Configurator Express』を参照してください。
- 標準コネクター・プロパティーの説明については、『標準コネクター・プロパティーの構成』、および 53 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。
- コネクター固有のプロパティーの詳細については、『コネクター固有プロパティーの構成』を参照してください。

標準コネクター・プロパティーの構成

標準構成プロパティーにより、すべてのコネクターによって使用される情報が提供されます。標準構成プロパティーの資料については、 53 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。これらのプロパティーの設定方法を説明したステップバイステップ手順については、 69 ページの『Connector Configurator Express の概要』を参照してください。

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

コネクター固有プロパティーの構成

コネクター固有の構成プロパティーは、コネクター・エージェントが実行時に必要とする情報を提供します。また、コネクター固有の構成プロパティーを使用すると、コネクター・エージェントのコード変更や再ビルドを行わなくても、エージェント内の静的情報またはロジックを変更できます。

表 10 に、コネクターのコネクター固有の構成プロパティーのリストを示します。プロパティーの説明については、以下の各セクションを参照してください。

表 10. コネクター固有のプロパティー

名前	指定可能な値	デフォルト値	必須
ArchiveDestination	*正常に処理されたメッセージのコピーが送信される宛先*		いいえ
ConfigurationMetaObject	*構成メタオブジェクト*		プロパティーの説明を参照
ConnectionFactoryName	*JNDI ストアで定義された JMS キューまたはトピックの接続ファクトリー*		はい
CTX_InitialContextFactory	*初期 JNDI コンテキストの設定に使用されるファクトリー・クラスの名前*		はい
CTX_ProviderURL	*接続ファクトリーが存在する JNDI コンテキストを示す URL*		はい
DataHandlerClassName	*インスタンスを生成するデータ・ハンドラー・クラスの名前*		プロパティーの説明を参照
DataHandlerConfigMO	*DataHandlerMimeType の構成情報を含むデータ・ハンドラー・メタオブジェクトの名前*	`MO_DataHandler_Default`	プロパティーの説明を参照
DataHandlerMimeType	*デフォルト・データ・ハンドラーの選択で使用する MIME タイプ*	`text/delimited`	プロパティーの説明を参照
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。*	`Create`	いいえ
ErrorDestination	*未処理メッセージの宛先*		いいえ

InDoubtEvents	`FailOnStartup`
	`Reprocess`
	`Ignore`
	`LogError`
		`Reprocess`	いいえ

InProgressDestination	*一時記憶域宛先*		いいえ
InputDestination	*ポーリング宛先の名前*		いいえ
LookupDestinationsUsingJNDI	`true` または `false`	`false`	いいえ
MessageFormatProperty	*メッセージ・フォーマットを指定するプロパティー名*	`JMSType`	いいえ
MessageResponseResultProperty	*要求された操作の結果を示す応答メッセージのプロパティー*	`WBI_Result`	はい (同期処理の場合)
PollQuantity	*InputDestination プロパティーで指定された各宛先で検索するメッセージの数*	`1`	いいえ
ReplyToDestination	*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宛先*		はい (同期処理の場合)
UnsubscribedDestination	*メッセージが認識されないか、またはそのメッセージがマップする対象のビジネス・オブジェクトがサポートされない場合に、インバウンド・メッセージのコピーが出力される宛先です。*		いいえ
UnsubscribeOnTerminate	*InputDestination から削除するトピックを指定します。*		いいえ

UseDefaults	`true`
	または
	`false`
		`false`	いいえ

UseDurableSubscriptions	`true`
	または
	`false`
		`false`	いいえ

ArchiveDestination
正常に処理されたメッセージのコピーが送信される宛先です。

デフォルト値は CWLD_ARCHIVE です。

ConfigurationMetaObject
コネクターの構成情報を含む静的なメタオブジェクトの名前です。

デフォルト値はありません。

ConnectionFactoryName
JMS プロバイダーとの接続を確立するときに、コネクターが検索し、使用する必要のある、JNDI ストアで定義された JMS キューまたはトピックの接続ファクトリーの名前です。この名前を検索する場合、コネクターは、CTX_InitialContextFactory および CTX_ProviderURL の各プロパティーによって設定された初期 JNDI コンテキストを使用します。

デフォルト = なし。

CTX_InitialContextFactory
初期 JNDI コンテキストの設定に使用されるファクトリー・クラスの名前です。

デフォルト = なし。

CTX_ProviderURL
接続要因が存在する JNDI コンテキストを示す完全修飾 URL です。この値はコンテキスト要因に渡されます。

デフォルト = 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。 DataHandlerConfigMO と DataHandlerMimeType の両方を指定するか、DataHandlerClassName のみを指定します。3 つのプロパティーすべてを指定しないでください。

注: 静的メタオブジェクト、または動的メタオブジェクト内の DataHandlerClassName 値は、このコネクターの構成プロパティーで指定される値よりも優先されます。メタオブジェクト内で DataHandlerClassName 値を指定しない場合、コネクターはコネクター構成プロパティーから値を取得します。

デフォルト = なし。

DataHandlerConfigMO
DataHandlerMimeType プロパティーで指定された MIME タイプの構成情報を含むメタオブジェクトの名前です。データ・ハンドラーの構成情報を提供します。DataHandlerConfigMO と DataHandlerMimeType を指定するか、DataHandlerClassName のみを指定します。3 つのプロパティーすべてを指定しないでください。

注: 静的メタオブジェクト、または動的メタオブジェクト内の DataHandlerConfigMO 値は、このコネクターの構成プロパティーで指定される値よりも優先されます。メタオブジェクト内で DataHandlerConfigMO 値を提供しない場合、コネクターはコネクター構成プロパティーから値を取得します。

デフォルト値は MO_DataHandler_Default です。

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。DataHandlerConfigMO と DataHandlerMimeType を指定するか、DataHandlerClassName のみを指定します。3 つのプロパティーすべてを指定しないでください。

注: 静的メタオブジェクト、または動的メタオブジェクト内の DataHandlerMimeType 値は、このコネクターの構成プロパティーで指定される値よりも優先されます。メタオブジェクト内で DataHandlerMimeType 値を指定しない場合、コネクターはコネクター構成プロパティーから値を取得します。

デフォルト = text/delimited

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

ErrorDestination

処理中にコネクターがエラーを検出したときに、インバウンド・メッセージのコピーが送信される宛先です。

デフォルト値は CWLD_ERROR です。

InDoubtEvents

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

- FailOnStartup: エラーをログに記録し、ただちにシャットダウンします。
- Reprocess: 残りのイベントを先に処理してから、入力キューのメッセージを処理します。
- Ignore: 進行中のキューのメッセージをすべて無視します。
- LogError: エラーをログに記録しますが、シャットダウンしません。

デフォルト値は Reprocess です。

注: InProgressDestination プロパティーを構成する場合は、このプロパティーの値を指定する必要があります。

InProgressDestination
処理中にメッセージが保留される一時的宛先です。

デフォルト = なし。

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

InputDestination プロパティーが指定されていない場合、コネクターはポーリングしません。

デフォルト = なし。

LookupDestinationsUsingJNDI
このプロパティーが true の場合、コネクターは、JNDI ストアのすべての JMS 宛先名を検索します。この場合、指定された宛先がすべて JNDI ストアで定義されている必要があります。

デフォルトで、コネクターはこのステップをスキップし、JMS プロバイダーが実行時に名前を適切な宛先へ変換することを許可します。

デフォルト = false

MessageFormatProperty
メッセージの入出力フォーマットを含む JMS メッセージのフィールドです。デフォルトで、コネクターは、インバウンド・メッセージの JMSType フィールドでメッセージ・フォーマットを調べ、そのメッセージ・フォーマットを、アウトバウンド・メッセージの JMSType フィールドに書き込みます。

デフォルト =JMSType

MessageResponseResultProperty
同期要求処理で必要とされます。このプロパティーは、コネクターが要求結果を確認するために検査する、応答 JMS メッセージのフィールドを指定します。このプロパティーは、非同期処理では使用されません。

デフォルト値は WBI_Result です。

PollQuantity
pollForEvents サイクル中に、InputDestination プロパティーで指定された各宛先で検索するメッセージの最大数です。

デフォルト値は 1 です。

ReplyToDestination

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宛先です。ターゲット・アプリケーションとの間での要求メッセージの交換を調整するために、コネクターが使用するデフォルトの宛先です。同期処理の場合にのみ、このプロパティーを指定します。

デフォルト = なし。

UnsubscribedDestination

メッセージが認識されないか、またはそのメッセージがマップする対象のビジネス・オブジェクトがサポートされない場合に、インバウンド・メッセージのコピーが出力される宛先です。このプロパティーが定義されており、有効な場合、コネクターは、アンサブスクライブされたメッセージのコピーをこの宛先に置きます。そうでない場合、メッセージは廃棄されます。

デフォルト = なし。

UnsubscribeOnTerminate

UserDurableSubscriptions が `true` に設定されている場合にのみ、適用できます。永続サブスクリプションを使用すると、コネクター構成からトピックを削除すると問題が発生します。コネクターが永続サブスクリプションを調べなくなっても、JMS プロバイダーはそのサブスクリプションのメッセージを保管し続けようとします。

InputDestination で指定されたリストからトピックを削除するときは常に、このプロパティー値で、削除するそれらのトピック (セミコロンで区切る) を指定します。既存の永続サブスクリプションを破棄するには、以下の手順を実行します。

1. サブスクリプションを終了する該当のトピック名を、InputDestination から UnsubscribeOnTerminate へと移動します。
2. コネクターを開始および停止します (これにより永続サブスクリプションが破棄されます)。
3. UnsubscribeOnTerminate に指定されたすべてのトピックを消去します。

この操作は InputDestination の値には影響ありません。

上記ステップを実行しなくてもコネクターには影響しませんが、JMS プロバイダーが、不必要なメッセージを保管するようになります。

デフォルト = なし。

UseDefaults

UseDefaults が `true` に設定されている場合、コネクターは、isRequired とマークされている各ビジネス・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検査します。

デフォルト = `false`

UseDurableSubscriptions

パブリッシュ/サブスクライブのトピック・スタイル・メッセージングにのみ、これを使用します。このプロパティーが `true` に設定されている場合、コネクターは、

該当する宛先に対して永続サブスクライバーとして働きます。コネクターは、オフラインの状態にあっても、JMS プロバイダーに対し、サブスクライブするトピックのメッセージをすべて保管するように指示します。これには大きなオーバーヘッドが伴います。コネクターは、オンラインの状態に戻ると、失ったすべての公表されたメッセージを再処理します。

デフォルト = false

メッセージ・スタイルの構成

アダプターは、JMS 規格で定義された point-to-point (PTP) メッセージングおよびパブリッシュ/サブスクライブ (Pub/Sub) メッセージングの両方のインターフェースをサポートします。アダプターが使用するメッセージング・スタイルは、コネクター固有プロパティー ConnectionFactoryName にユーザーが指定する管理オブジェクトのタイプによって決まります。以下の手順に進む前に、23 ページの『ConnectionFactoryName』を参照してください。

- 『PTP メッセージ・スタイルの構成』
- 『Pub/Sub スタイルの構成』

PTP メッセージ・スタイルの構成

PTP メッセージ・スタイルでアダプターのインスタンスを構成するには、次のようにします。

1. 「Connector Configurator Express」を開きます。
2. 「コネクター固有プロパティー」タブをクリックします。
3. JNDI ストアの JMS *Queue*ConnectionFactory のインスタンスにマップする ConnectionFactoryName の名前を指定します。アダプターは PTP スタイルで動作し、宛先を示すすべてのコネクターおよびメタオブジェクトのプロパティー (OutputDestination プロパティーなど) が、キューを表すものと見なします。

Pub/Sub スタイルの構成

Pub/Sub メッセージ・スタイルでアダプターのインスタンスを構成するには、次のようにします。

1. 「Connector Configurator Express」を開きます。
2. 「コネクター固有プロパティー」タブをクリックします。
3. JNDI ストアの JMS *Topic*ConnectionFactory のインスタンスにマップする ConnectionFactoryName の名前を指定します。アダプターはパブリッシュ/サブスクライブ・スタイルで動作し、宛先を示すすべてのコネクターおよびメタオブジェクトのプロパティー (OutputDestination プロパティーなど) が、トピックを表すものと見なします。

JNDI の構成

JMS プロバイダーへの接続を確立するには、コネクターは、JMS 接続ファクトリーへアクセスする必要があります。JMS はファクトリーのインターフェースを定義しています。ただし、各 JMS プロバイダーは、独自のインプリメンテーションを提供する必要があります。コネクターにこのファクトリー・インプリメンテーション

への参照が作成されると、コネクターは、プロバイダーの専有プロトコルまたは ID の知識がなくても、JMS プロバイダーとの接続を確立し、通信できるようになります。

コネクターを移植可能にするには、接続ファクトリーを JNDI ストアに置く必要があります。ユーザーまたはシステム管理者は、インプリメンテーションのときに、接続ファクトリーの作成と構成を行い、それをユーザー定義名で JNDI ストアに置く必要があります。実行時に、コネクターは、JNDI ストアとの接続を確立し、接続ファクトリーを検索し、それを使用して JMS プロバイダーへの接続を確立します。

接続ファクトリーまたはユーザーが作成するその他の管理 JMS オブジェクトを含む、独自の JNDI インプリメンテーションを提供する JMS プロバイダーもあります。この方法を使用すると、ユーザーは JMS アダプターを非常に簡単に構成することができます。その他の JMS プロバイダーでは、ユーザーは、外部 JNDI プロバイダーのインストールと構成を行い、接続ファクトリーを作成し、それをアダプターが使用できるようにしなければならない場合があります。詳細については JNDI プロバイダーの資料を参照してください。

JNDI の環境変数および構成の詳細については、`www.javasoft.com` を参照してください。JNDI (MA88 パッチ適用済み) の構成の詳細については、『WebSphere MQ Java クライアント・ライブラリーを使用した JNDI の構成』を参照してください。

WebSphere MQ Java クライアント・ライブラリーを使用した JNDI の構成

WebSphere MQ Java クライアント・ライブラリーを使用した JNDI の構成方法の解説については、85 ページの『キュー・ベース・メッセージングの構成』および 86 ページの『トピック・ベース・メッセージングの構成』を参照してください。

メタオブジェクトの構成

Connector for JMS は、2 種類のメタオブジェクトを認識および読み取ることができます。

- 静的なコネクター・メタオブジェクト
- 動的な子メタオブジェクト

動的な子メタオブジェクトの属性値は、静的なメタオブジェクトの属性値と重複し、それらをオーバーライドします。メタデータ、および静的メタオブジェクトと動的メタオブジェクトとの比較の概要については、7 ページの『メタデータおよびメタオブジェクト』を参照してください。

どのメタオブジェクトが使用しているインプリメンテーションに最適かを判別するには、以下のことを検討してください。

- **静的メタオブジェクト**
  - 各メッセージに対するメタデータがすべて決まっていて、構成時に指定できる場合に有効です。

- ビジネス・オブジェクト・タイプによって値を指定するよう制限されます。例えば、Customer タイプ・オブジェクトはすべて同じ宛先に送信する必要があります。

- **動的メタオブジェクト**
  - メッセージ・ヘッダーの情報に対し、ビジネス・プロセス・アクセスを提供します。
  - ビジネス・タイプに関係なく、ビジネス・プロセスが、実行時にメッセージの処理を変更できるようにします。例えば、動的メタオブジェクトを使用すると、アダプターに送られる各 Customer タイプ・オブジェクトに別々の宛先を指定できます。
  - サポートされるビジネス・オブジェクトの構造を変更する必要があります。そのような変更では、マップおよびビジネス・プロセスを変更しなければならない場合があります。
  - カスタム・データ・ハンドラーを変更する必要があります。

メタオブジェクト・プロパティー

表 11 は、メタオブジェクトでサポートされるプロパティーの完全なリストです。メタオブジェクトをインプリメントする場合は、これらのプロパティーを参照してください。

両方のオブジェクトですべてのプロパティーを使用できるわけではありません。メッセージ・ヘッダーとの間で、すべてのプロパティーが読み取り可能または書き込み可能であるわけでもありません。コネクターが特定のプロパティーをどのように解釈および使用するかを確認するには、1 ページの『第 1 章 Adapter for JMS の概要』の、イベントおよび要求の処理に関する該当セクションを参照してください。

表 11. JMS メタオブジェクト・プロパティー

プロパティー名	静的メタオブジェクトで定義可能	動的メタオブジェクトで定義可能	説明
DataHandlerConfigMO	はい	はい	構成情報を提供するために、データ・ハンドラーに渡されるメタオブジェクト。静的なメタオブジェクトに指定された場合、この値は DataHandlerConfigMO コネクター・プロパティーに指定された値をオーバーライドします。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ーを使用します。データ形式が実際のビジネス・データに依存する可能性がある場合は、要求処理には動的な子メタオブジェクトを使用します。指定されたビジネス・オブジェクトはコネクター・エージェントによりサポートされていることが必要です。21 ページの『コネクター固有プロパティーの構成』 の説明を参照してください。

表 11. JMS メタオブジェクト・プロパティー (続き)

プロパティー名	静的メタオブジェクトで定義可能	動的メタオブジェクトで定義可能	説明
DataHandlerMimeType	はい	はい	使用すると、特定の MIME タイプに基づいたデータ・ハンドラーを要求できます。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・プロパティー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ーを使用します。データ形式が実際のビジネス・データに依存する可能性がある場合は、要求処理には動的な子メタオブジェクトを使用します。DataHandlerConfigMO に指定されたビジネス・オブジェクトは、このプロパティーの値に対応する属性を含める必要があります。 21 ページの『コネクター固有プロパティーの構成』 の説明を参照してください。
DataHandlerClassName	はい	はい	21 ページの『コネクター固有プロパティーの構成』 の説明を参照してください。
InputFormat	はい	はい	インバウンド (イベント) メッセージのフォーマットまたはタイプ。この値は、メッセージ内容の識別を支援します。メッセージを生成したアプリケーションによって指定されます。メッセージ・フォーマットの定義でコネクターが考慮するフィールドは、コネクター固有プロパティー MessageFormatProperty によって、ユーザーが定義できます。
OutputFormat	はい	はい	アウトバウンド・メッセージで読み込まれるフォーマット。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。
InputDestination	はい	はい	このプロパティーは、着信メッセージをビジネス・オブジェクトとマッチングする目的のみで使用されます。対照的に、InputDestination のコネクター固有のプロパティーは、アダプターがポーリングする宛先を定義します。これはポーリングする宛先を判別するためにアダプターが使用する唯一のプロパティーです。 MO 内では、InputDestination プロパティーおよび InputFormat プロパティーは、アダプターが任意の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基準として機能します。この機能をインプリメントするためには、コネクター固有のプロパティーを使用して複数の入力宛先を構成し、オプションで、着信メッセージの入力フォーマットに基づき異なるデータ・ハンドラーをそれぞれにマップします。 このプロパティーは、デフォルトの型変換プロパティーを使用して設定しないでください。値は以下で使用されます。
OutputDestination	はい	はい	アウトバウンド・メッセージが書き込まれる宛先。

表 11. JMS メタオブジェクト・プロパティー (続き)

プロパティー名	静的メタオブジェクトで定義可能	動的メタオブジェクトで定義可能	説明
ResponseTimeout	はい	はい	同期要求処理の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。このプロパティーが定義されていないかまたはゼロより小さい値が設定されている場合、コネクターは応答を待機せずにすぐに SUCCESS を戻します。
TimeoutFatal	はい	はい	同期要求処理において、応答が受信されない場合にコネクターにエラー・メッセージを戻させるために使用されます。このプロパティーが True の場合、応答が ResponseTimeout で指定した時間内に受信されなければ、コネクターはブローカーに APPRESPONSETIMEOUT を戻します。このプロパティーが未定義か、または False にセットされている場合、応答がタイムアウトになるとコネクターは要求をエラーにしますが、終了はしません。デフォルト = False。
JMS メッセージ・ヘッダーに限定してマッピングされるフィールドを以下に示します。説明、値の解釈などについては、JMS API 仕様を参照してください。JMS プロバイダーは、一部のフィールドについて、異なった解釈をする場合があります。それらの違いについて、JMS プロバイダーの資料も確認してください。			
ReplyToDestination		はい	要求の応答メッセージが送られる宛先。
Type		はい	メッセージのタイプ。JMS プロバイダーによって異なりますが、一般には、ユーザーによる定義が可能。
MessageID		はい	メッセージの固有 ID (JMS プロバイダーに固有)。
CorrelationID	はい	はい	この応答を開始した要求メッセージの ID を示すために、応答メッセージで使用されます。
Delivery Mode	はい	はい	MOM システムでメッセージを永続させるかどうかを指定します。許容値は以下のとおりです。 1=非永続 2=永続 JMS プロバイダーによっては、その他の値を使用できます。
Priority		はい	メッセージの優先順位 (数値)。許容値は、0 から 9 です (数値が小さいほど優先順位は高くなる)。
Destination		はい	MOM システムでの、メッセージの現在または最後の (削除された場合) 場所。
Expiration		はい	メッセージの存続時間。ゼロが指定されると、有効期限がゼロに設定されます。ゼロを指定すると JMS プロバイダーは、メッセージを有効期限切れにしません。
Redelivered		はい	JMS プロバイダーが以前にクライアントへのメッセージ配信を試みた可能性は高いが、受取が確認されなかったことを示します。
Timestamp		はい	時間メッセージが JMS プロバイダーに渡されました。
UserID		はい	メッセージを送信するユーザーの ID。
AppID		はい	メッセージを送信するアプリケーションの ID。
DeliveryCount		はい	配信を試みる回数。
GroupID		はい	メッセージ・グループの ID。
GroupSeq		はい	グループ ID で指定されたメッセージ・グループにおける、このメッセージの順序。

表 11. JMS メタオブジェクト・プロパティー (続き)

プロパティー名	静的メタオブジェクトで定義可能	動的メタオブジェクトで定義可能	説明
JMSProperties		はい	37 ページの『JMS プロパティー』を参照してください。

静的メタオブジェクトの構成

JMS 構成の静的なメタオブジェクトには、さまざまなビジネス・オブジェクトに定義された変換プロパティーのリストが含まれます。静的メタオブジェクトのサンプルを参照するには、Business Object Designer Express を起動し、アダプターと一緒に出荷される、次のサンプルを開きます。

`connectors¥JMS¥Samples¥Sample_JMS_MO_Config.xsd`

コネクターは、いつでも、最大で 1 つの静的メタオブジェクトをサポートします。静的メタオブジェクトは、コネクター・プロパティー ConfigurationMetaObject で名前を指定してインプリメントします。

静的メタオブジェクトは、ビジネス・オブジェクトと動詞の単一の組み合わせ、およびそのオブジェクトの処理に関連したすべてのメタデータを、各属性が表す構造になっています。各属性の名前は、Customer_Create のように、ビジネス・オブジェクト・タイプと動詞の名前を下線で区切ったものです。属性のアプリケーション固有情報は、セミコロンで区切られた 1 つ以上の名前と値のペアで構成され、この固有なオブジェクトと動詞の組み合わせに指定するメタデータ・プロパティーを表します。

表 12. 静的メタオブジェクトの構造

属性名	アプリケーション固有のテキスト
`<business object type>_<verb>`	`property=value;property=value;...`
`<business object type>_<verb>`	`property=value;property=value;...`

以下のメタオブジェクトを例にとります。

表 13. 静的メタオブジェクト構造のサンプル

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

このサンプルのメタオブジェクトは、コネクターが、タイプ Customer に動詞 Create が付いた要求ビジネス・オブジェクトを受け取った場合、それをフォーマット CUST のメッセージに変換し、宛先 QueueA に置くことを知らせます。カスタマー・オブジェクトが動詞 Update を持つ場合、メッセージは QueueB に置かれます。オブジェクト・タイプが Order であり、動詞 Create を持つ場合、コネクターは、そのオブジェクトを、フォーマット ORDER で変換し、QueueC に配信します。コネクターに渡されるその他のビジネス・オブジェクトは、アンサブスクライブされているものとして処理されます。

オプションで、1 つの属性 Default を指定し、それに ASI の 1 つ以上のプロパティーを割り当てることができます。メタオブジェクトに含まれるすべての属性で、デフォルト属性のプロパティーは、特定のオブジェクトと動詞属性のプロパティーに結合されます。これは、全体に適用する 1 つ以上のプロパティーがある (オブジェクトと動詞の組み合わせに関係なく) 場合に便利です。以下の例の場合、コネクターは、Customer_Create および Order_Create のオブジェクトと動詞の組み合わせが、それらの個別のメタデータ・プロパティー以外に、OutputDestination=QueueA を持つとみなします。

表 14. 静的メタオブジェクト構造のサンプル

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

29 ページの『メタオブジェクト・プロパティー』 の 29 ページの表 11 では、静的メタオブジェクトで、アプリケーション固有情報として指定できるプロパティーについて説明しています。

静的メタオブジェクトをインプリメントするには、以下のようにします。

1. Business Object Designer Express を起動します。詳細については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。
2. サンプルのメタオブジェクト connectors¥JMS¥Samples¥Sample_JMS_MO_Config.xsd を開きます。図 3 に Business Object Designer Express 内のサンプルの静的メタオブジェクトを示します。

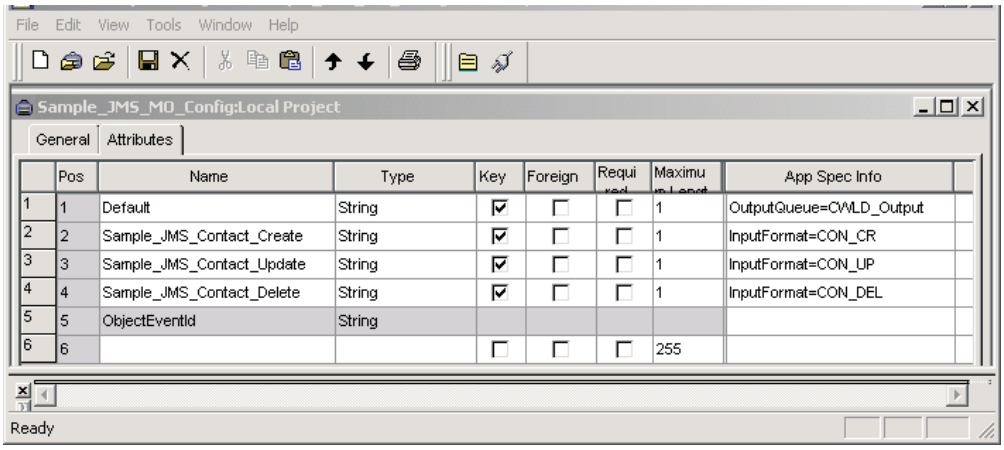


図 3. 静的メタオブジェクトのサンプル

3. 29 ページの表 11 を参照して、要件に適合するように属性および ASI を編集してから、メタオブジェクト・ファイルを保管します。
4. このメタオブジェクト・ファイルの名前を、コネクター・プロパティー ConfigurationMetaObject の値として指定します。

入力宛先へのデータ・ハンドラーのマッピング

静的メタオブジェクトのアプリケーション固有情報で InputDestination プロパティーを使用することにより、データ・ハンドラーと入力宛先を関連付けることができます。この機能は、異なる書式や変換要件を持つ複数の取引先と取り引きする場合に役立ちます。

データ・ハンドラーを入力宛先にマップするには、以下のようにします。

1. Connector Configurator Express を起動します。詳細については、69 ページの『付録 B. Connector Configurator Express』を参照してください。
2. コネクター固有プロパティー (25 ページの『InputDestination』を参照) を使用して、1 つ以上の入力宛先を構成します。複数の宛先名はセミコロンで区切る必要があります。
3. それぞれの入力宛先ごとに、宛先 (PTP メッセージング・スタイルをインプリメントしている場合はキュー・マネージャー) および入力宛先名を指定し、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 タイプを指定します。

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

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

動的子メタオブジェクトの構成

静的な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、コネクターは、ビジネス・オブジェクト・インスタンスごとに実行時に配信されたメタデータをオプションで受け入れることができます。

動的メタオブジェクトを使用すると、要求処理のときに、要求ごとに、コネクターが使用するメタデータを変更してビジネス・オブジェクトを処理できます。また、イベント処理のときに、イベント・メッセージの情報を検索することができます。

動的メタオブジェクトは、各属性が、次のような単一のメタデータのプロパティーと値を表す構造になっています。`meta-object property name =meta-object property value`

動的メタオブジェクトをインプリメントするには、それを子としてトップレベル・オブジェクトに追加し、トップレベル・オブジェクト ASI に名前と値のペア `cw_mo_conn=<MO attribute>` を組み込みます。*<MO attribute>* は、動的メタオブジェクトを表すトップレベル・オブジェクトの属性名です。以下に例を示します。

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

上記のように指定された要求を受け取ると、コネクターは、Customer オブジェクトを CUST というフォーマットを持ったメッセージに変換してから、メッセージをキュー QueueA に置きます。

ビジネス・オブジェクトは、同じ動的メタオブジェクトでも、異なった動的メタオブジェクトでも使用できます。また、まったく使用しないことも可能です。

注: すべての標準 IBM WebSphere データ・ハンドラーは、 cw_mo_ タグを認識することによって、この動的メタオブジェクト属性を無視するように設計されています。アダプターに使用するカスタム・データ・ハンドラーを開発する場合、同じようにする必要があります。

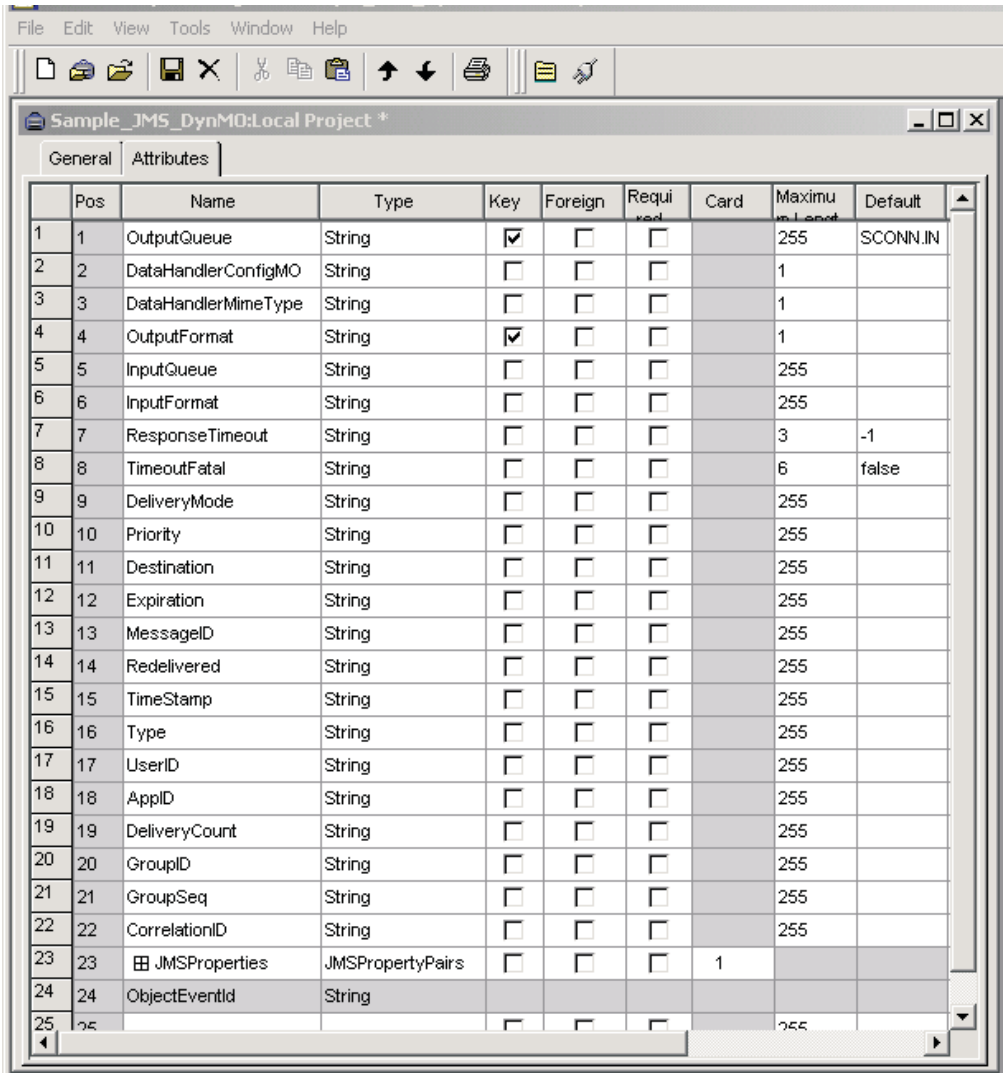
コネクターは、コネクターに渡されるトップレベル・ビジネス・オブジェクトに子として追加される動的なメタオブジェクトから、変換プロパティーを認識し、読み取ります。この動的な子メタオブジェクトの属性値は、コネクターの構成に使用される静的なメタオブジェクトに指定可能であった変換プロパティーと重複します。

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

29 ページの『メタオブジェクト・プロパティー』 の 29 ページの表 11 では、動的メタオブジェクトで、アプリケーション固有情報として指定できるプロパティーについて説明します。

動的メタオブジェクトを構成するには、以下のようにします。

1. Business Object Designer Express を起動します。詳細については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。
2. サンプルのメタオブジェクト connectors¥JMS¥Samples¥Sample_JMS_DynMO.xsd を開きます。 36 ページの図 4 に Business Object Designer Express 内のサンプルの動的メタオブジェクトを示します。



	Pos	Name	Type	Key	Foreign	Requi red	Card	Maximu m Lengt	Default
1	1	OutputQueue	String	☑	☐	☐		255	SCONN.IN
2	2	DataHandlerConfigMO	String	☐	☐	☐		1	
3	3	DataHandlerMimeType	String	☐	☐	☐		1	
4	4	OutputFormat	String	☑	☐	☐		1	
5	5	InputQueue	String	☐	☐	☐		255	
6	6	InputFormat	String	☐	☐	☐		255	
7	7	ResponseTimeout	String	☐	☐	☐		3	-1
8	8	TimeoutFatal	String	☐	☐	☐		6	false
9	9	DeliveryMode	String	☐	☐	☐		255	
10	10	Priority	String	☐	☐	☐		255	
11	11	Destination	String	☐	☐	☐		255	
12	12	Expiration	String	☐	☐	☐		255	
13	13	MessageID	String	☐	☐	☐		255	
14	14	Redelivered	String	☐	☐	☐		255	
15	15	TimeStamp	String	☐	☐	☐		255	
16	16	Type	String	☐	☐	☐		255	
17	17	UserID	String	☐	☐	☐		255	
18	18	AppID	String	☐	☐	☐		255	
19	19	DeliveryCount	String	☐	☐	☐		255	
20	20	GroupID	String	☐	☐	☐		255	
21	21	GroupSeq	String	☐	☐	☐		255	
22	22	CorrelationID	String	☐	☐	☐		255	
23	23	⊞ JMSProperties	JMSPropertyPairs	☐	☐	☐	1		
24	24	ObjectEventId	String						
25				☐	☐	☐		255	

図 4. 動的メタオブジェクトのサンプル

3. このビジネス・オブジェクトの要件に適合するように属性およびプロパティーを編集し、保管します。
4. 動的メタデータ・オブジェクトを子としてトップレベル・オブジェクトに追加し、トップレベル・オブジェクト ASI に名前と値のペア `cw_mo_conn=<MO attribute>` を組み込みます。*<MO attribute>* は、動的メタオブジェクトを表すトップレベル・オブジェクトの属性名です。

ポーリング中の動的な子メタオブジェクトの含まれるデータ

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

37 ページの表 15 に、動的な子メタオブジェクトがポーリング用に構造化される方法を示します。

表 15. ポーリング用の JMS 動的子メタオブジェクト構造

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

表 15 に示すように、動的子メタオブジェクトで追加の属性 Input_Format および Inputdestination を定義できます。 Input_Format は検索したメッセージのフォーマットで読み込まれ、InputDestination 属性には特定のメッセージの検索先となる宛先の名前が含まれます。子メタオブジェクト内にこれらのプロパティーが定義されていない場合、これらには値が取り込まれません。

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 InputDestination 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公表します。

JMS ヘッダーおよび動的子メタオブジェクト属性

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

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

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

- Int
- Float
- Double
- Long
- Short
- Byte

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

表 16. JMS プロパティー属性のアプリケーション固有情報

属性	指定可能な値	ASI	コメント
Name	有効な JMS プロパティー名 (有効 = ASI で定義されたタイプと互換性がある)	name=<*JMS プロパティー名*>;type=<*JMS プロパティー・タイプ*>	ベンダーによっては、拡張機能を提供するために特定のプロパティーを予約している場合があります。一般に、ユーザーはベンダー固有の機能にアクセスする場合以外は、JMS で開始するカスタム・プロパティーを定義してはなりません。
Type	String	type=<コメントを参照>	これは JMS プロパティーのタイプです。JMS API は、JMS メッセージに値を設定するための多くのメソッドを提供します (例: setIntProperty、setLongProperty、setStringProperty)。ここで指定する JMS プロパティーのタイプによって、どのメソッドを使用してメッセージのプロパティー値を設定するかが決まります。

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

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

```
|-- OutputDestination = QueueA
|-- JMSProperties
     |-- RoutingCode = 123 (ASI= name=RoutingCode;type=Int)
     |-- Dept = FD (ASI= name=RoutingDept;type=String)
```

別の例を示すために、図 5 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 つのプロパティー (ID、GID、RESPONSE、および RESPONSE_PERSIST) の定義を示します。属性のアプリケーション固有情報はそれぞれの名前およびタイプを定義します。例えば、属性 ID はタイプ String の JMS プロパティー ID にマップされます。

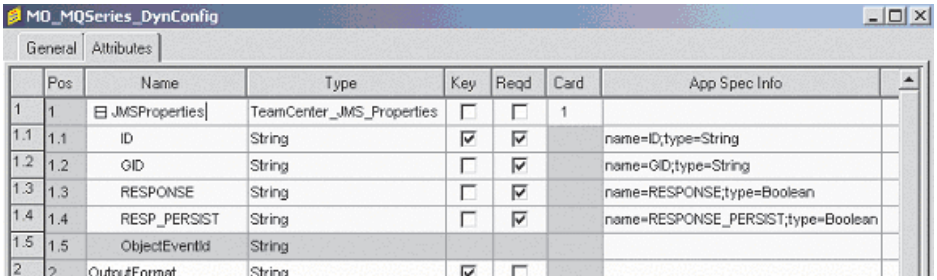


図 5. 動的メタオブジェクトの JMS プロパティー属性

開始スクリプトの構成

コネクターとともに提供されるのは、JMS.batstartup script です。コネクターの始動、コネクターの停止、およびコネクターの一時始動ログ・ファイルについては、「*WebSphere Business Integration Server Express インストール・ガイド*」の始動に関する章を参照してください。

複数のコネクター・インスタンスの作成

注: このアダプター (あるいは WebSphere Business Integration Server Express または Express Plus に付属の任意のアダプター) の追加インスタンスを作成すると、そのアダプター・インスタンスは、配置できるアダプターの総数を制限するライセンス機能によって、別のアダプターとしてカウントされます。

以下に示すステップを実行することによって、コネクターの複数のインスタンスを作成して実行するように、ご使用のシステムを設定することができます。次のようにする必要があります。

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

新規ディレクトリーの作成

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

- Windows プラットフォームの場合:

  ```
  ProductDir¥connectors¥connectorInstance
  ```

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

  ```
  ProductDir¥repository¥connectorInstance
  ```

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

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

- OS/400 プラットフォームの場合:

  ```
  /QIBM/UserData/WBIServer43/WebSphereICSName/connectors/connectorInstance
  ```

  ここで、*connectorInstance* はコネクター・インスタンスを固有に識別し、*WebSphereICSName* はコネクターの実行に使用する InterChange Server Express インスタンスの名前です。

  コネクターに、コネクター固有のメタオブジェクトがある場合、コネクター・インスタンス用のメタオブジェクトを作成する必要があります。メタオブジェクトをファイルとして保管する場合は、次のディレクトリーを作成して、ファイルを `/QIBM/UserData/WBIServer43/WebSphereICSName/repository/connectorInstance` に格納します。

  ここで、*WebSphereICSName* はコネクターの実行に使用する InterChange Server Express インスタンスの名前です。

- Linux プラットフォームの場合:

  ```
  ProductDir/connectors/connectorInstance
  ```

  ここで *connectorInstance* は、コネクター・インスタンスを一意的に示します。コネクターに、コネクター固有のメタオブジェクトがある場合、コネクター・インスタンス用のメタオブジェクトを作成する必要があります。メタオブジェクトをファイルとして保管する場合は、次のディレクトリーを作成して、ファイルを `ProductDir/repository/connectorInstance` に格納します。

  InterChange ServerExpress サーバー名は、例えば `connector_manager -start connName WebSphereICSName [-cConfigFile]` のように、`connector_manager` のパラメーターとして指定できます。

ビジネス・オブジェクト定義の作成

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

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

     ¥ProductDir¥repository¥initialConnectorInstance

     作成した追加ファイルは、¥ProductDir¥repository の適切なコネクター・インスタンス・サブディレクトリーに置かれていなければなりません。
   - OS/400: /QIBM/UserData/WBIServer43/WebSphereICSName/repository/initialConnectorInstance

     ここで、*WebSphereICSName* はコネクターの実行に使用する InterChange Server Express インスタンスの名前です。

     作成した追加ファイルは、/QIBM/UserData/WBIServer43/WebSphereICSName/repository の適切な connectorInstance サブディレクトリー内に存在している必要があります。
   - Linux:

     /ProductDir/repository/initialConnectorInstance

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

コネクター定義の作成

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

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

始動スクリプトの作成

始動スクリプトは以下のように作成します。

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

   dirname

(Linux の場合のみ) 始動スクリプト CONJAR を CONJAR=${CONDIR}/CW${CONNAME}.jar から「CONJAR=${CONDIR}/CWJMS.jar」に変更する必要があります。

2. この始動スクリプトを、 40 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. (Windows の場合のみ) 始動スクリプトのショートカットを作成します。
4. (Windows の場合のみ) 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。
5. (OS/400 の場合のみ) 次の情報を使用してコネクターのジョブ記述を作成します: CRTDUPOBJ OBJ(QWBIJMSC) FROMLIB(QWBISVR43) OBJTYPE(*JOBD) TOLIB(QWBISVR43) NEWOBJ(new-jobd-name)

   ここで、*new-jobd-name* は、新規 JMS コネクターのジョブ記述で使用する 10 文字の名前です。
6. (OS/400 の場合のみ) 新規コネクターをコンソールに追加します。コンソールの詳細については、コンソールに付属のオンライン・ヘルプを参照してください。

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

コネクターの始動

コネクターは、**コネクター始動スクリプト**を使用して明示的に始動する必要があります。始動スクリプトは、次に示すようなコネクターのランタイム・ディレクトリーに存在していなければなりません。例えば、Windows の場合は以下を使用します。

ProductDir¥connectors¥*connName*

ここで、*connName* はコネクターを示します。始動スクリプトの名前は、表 17 に示すように、オペレーティング・システム・プラットフォームによって異なります。

表 17. コネクターの始動スクリプト

オペレーティング・システム	始動スクリプト
Windows	start_*connName*.bat
OS/400	start_connName.sh
Linux	connector_manager -start connName[-cConfigFile] により、環境変数が設定され、start_connName.sh 始動スクリプトが自動的に開始されます。始動スクリプトを手動で実行する必要はありません。

コマンド行の始動オプションなどのコネクターの始動方法の詳細については、「システム*管理ガイド*」を参照してください。

始動スクリプトの起動 (Windows の場合)

Windows プラットフォームでは、以下の方法でコネクターの始動スクリプトを起動できます。

- 「**スタート**」メニューから
  - 「**プログラム**」>「**IBM WebSphere Business Integration Express**」>「**アダプター**」>「**コネクター**」>「**ご使用のコネクター名**」を選択します。デフォルトでは、プログラム名は「IBM WebSphere Business Integration Express」となっています。ただし、これはカスタマイズすることができます。あるいは、ご使用のコネクターへのデスクトップ・ショートカットを作成することもできます。
  - Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムがブートしたとき (自動サービスの場合)、または Windows サービス・ウィンドウを通じてサービスを始動したとき (手動サービスの場合) に、コネクターが始動します。
- コマンド行から

  `start_connName connName WebSphereICSName [-cconfigFile ]`

  ここで、*connName* はコネクターの名前であり、*WebSphereICSName* は InterChange Server Express インスタンスの名前です。デフォルトでは、InterChange Server Express インスタンスの名前は WebSphereICS です。

始動スクリプトの起動 (OS/400 の場合)

OS/400 プラットフォームでは、以下の方法でコネクターの始動スクリプトを起動できます。

- Windows から

  WebSphere Business Integration Server Express Console がインストールされているマシンから、「プログラム」>「IBM WebSphere Business Integration Console」>「コンソール」を選択します。次に、OS/400 システム名または IP アドレス、および *JOBCTL 特殊権限を持つユーザー・プロファイルとパスワードを指定します。アダプターのリストから connName アダプターを選択し、「アダプターを始動」ボタンを選択します。
- OS/400 コマンド行から

  バッチ・モード: CL コマンド QSH を実行し、QSHELL 環境から次のコマンドを実行する必要があります。
  `/QIBM/ProdData/WBIServer43/bin/submit_adapter.sh connName WebSphereICSName pathToConnNameStartScript jobDescriptionName`
  ここで、*connName* はコネクター名であり、*WebSphereICSName* は InterChange Server Express サーバー名 (デフォルトは QWBIDFT)、*pathToConnNameStartScript* はコネクターの始動スクリプトの絶対パス、*jobDescriptionName* は QWBISVR43 ライブラリーで使用するジョブ記述の名前です。

  対話モード: CL コマンド QSH を実行し、QSHELL 環境から次のコマンドを実行する必要があります。
  `/QIBM/UserData/WBIServer43/WebSphereICSName/connectors/connName/start_connName.sh connName WebSphereICSName [-cConfigFile]`
  ここで、*connName* はご使用のコネクターの名前であり、*WebSphereICSName* は Interchange Server Express インスタンスの名前です。

注: TCP/IP サーバーを使用して始動するには、コマンド `/QIBM/ProdData/WBIServer43/bin/add_autostart_adapter.sh` を使用します。このとき、上記で説明したように、パラメーター connName、WebSphereICSName、pathToConnNameStartScript、および jobDescriptionName をこの順序で指定します。

始動スクリプトの起動 (Linux の場合)

Linux プラットフォームでは、以下の方法でコネクターの始動スクリプトを起動できます。

- `connector_manager -start` *`connName`* `WebSphereICSName [-`*`cConfigFile`*`]`

  ここで、*connName* はコネクターの名前であり、*WebSphereICSName* は InterChange Server Express インスタンスの名前です。

コネクターの停止

コネクターを停止する方法は、コネクターが始動された方法によって異なります。

コネクターの停止 (Windows から)

Windows プラットフォームでは、以下の方法でコネクターを停止できます。

- コネクター・ウィンドウをアクティブ化し、「q」と入力します。
- コネクターが Windows サービスとして始動した場合は、コントロール・パネルから、
  「コントロール パネル」>「管理ツール」>「サービス」>「CWConnectorWBIJMSAdapter」を選択して停止できます。

コネクターの停止 (OS/400 から)

OS/400 プラットフォームでは、以下の方法でコネクターを停止できます。

- コンソールまたはコマンド行から

  コンソールを使用して、または OS/400 コマンド入力から QSHELL で「submit_adapter.sh」スクリプトを使用してコネクターを始動した場合は、CL コマンド *WRKACTJOB SBS(QWBISVR43)* を使用して Server Express 製品に対するジョブを表示します。リストをスクロールして、コネクターのジョブ記述に一致するジョブ名を持つジョブを探し出します。例えば、JMS コネクターの場合のジョブ名は QWBIJMSC です。

  「コントロール パネル」このジョブに対してオプション 4 を選択し、F4 を押して ENDJOB コマンドのプロンプトを取得します。次に、オプション・パラメーターとして *IMMED を指定し、Enter を押します。
- Windows システムでは、始動スクリプトを起動すると、そのコネクター用の別個の「コンソール」ウィンドウが作成されます。このウィンドウで、「Q」と入力して Enter キーを押すと、コネクターが停止します。
- QSHELL から start_connName.sh スクリプトを使用してアダプターを始動した場合は、F3 を押してコネクターを終了します。

コネクターの停止 (Linux から)

- Linux システムでは、コネクターはバックグラウンドで実行されるので、個別のウィンドウはありません。代わりに、以下のコマンドを実行してコネクターを停止します。

  ```
  connector_manager connName -stop serverName
  ```

  ここで、*connName* はコネクターの名前であり、*serverName* は InterChange Server Express インスタンスの名前です。

第 3 章 ビジネス・オブジェクトの作成または変更

- 『コネクターのビジネス・オブジェクトの構造』

コネクターとともに提供されるのは、サンプルのビジネス・オブジェクトのみです。システム・インテグレーター、コンサルタント、またはカスタマーは、ビジネス・オブジェクトをビルドする必要があります。

この章では、ビジネス・オブジェクトのコネクター要件について説明します。これらの情報は、新規のビジネス・オブジェクトをインプリメントするための手引きとして役立ちます。

コネクターのビジネス・オブジェクトの構造

コネクターのインストールが完了したら、ビジネス・オブジェクトを作成する必要があります。ビジネス・オブジェクトの構造については、構成済みのデータ・ハンドラーによって定められている以外の要件はありません。コネクターが処理するビジネス・オブジェクトには、WebSphere InterChange Server Express で許可された任意の名前を付けることができます。

コネクターは宛先からメッセージを検索し、ビジネス・オブジェクト (メタオブジェクトによって定義されたもの) にメッセージの内容を読み込もうとします。厳密に言うと、コネクターはビジネス・オブジェクト構造を制御したり、それに影響を及ぼしたりはしません。そのような働きは、メタオブジェクト定義とコネクターのデータ・ハンドラー要件の機能です。実際に、ビジネス・オブジェクト・レベルのアプリケーション・テキストは存在しません。むしろ、ビジネス・オブジェクトの検索や受け渡しの際のコネクターの主な役割は、ビジネス・オブジェクトへのメッセージ (逆もまた同様) の処理をモニターしてエラーの有無を確認することです。

ビジネス・オブジェクトの作成

1. InterChange Server Express を JMS アダプターのインスタンスで構成するときに、InterChange Server Express がビジネス・オブジェクトを送信するアプリケーションを指定し、それを構成します。
2. JMS 宛先のメッセージをターゲット・アプリケーションによる処理に適したビジネス・オブジェクトに変換できるデータ・ハンドラーで、コネクターを構成します。これを行うには、DataHandlerConfigMO および DataHandlerMimeType のコネクター・プロパティーを指定するか、DataHandlerClassName プロパティーを指定します。詳細については、21 ページの『コネクター・プロパティーの構成』を参照してください。オプションとして、静的および動的メタオブジェクトで、特別なデータ・ハンドラー処理規則を指定できます。詳細については、29 ページの『メタオブジェクト・プロパティー』を参照してください。
3. アプリケーション固有のビジネス・オブジェクトを作成するには、Business Object Designer Express を使用します。詳細については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。
4. 作成したビジネス・オブジェクトをサポートされているビジネス・オブジェクトに追加します。Connector Configurator Express を使用して、JMS 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、作成したビジネス・オブジェクトを追加し、**Message Set ID** をサポートされている各ビジネス・オブジェクトの固有な値に設定します。Connector Configurator Express を使用してサポートされているビジネス・オブジェクトを追加する方法の詳細については、78 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』を参照してください。

第 4 章 トラブルシューティング

この章では、コネクターがエラーおよびトレースを処理する方法、およびコネクターの始動時および実行時に発生する可能性がある問題について説明します。

エラー処理

コネクターが生成するすべてのエラー・メッセージは、JMSConnector.txt という名前のメッセージ・ファイルに保管されます (ファイル名は、LogFileName 標準コネクター構成プロパティーによって決定されます)。それぞれのエラー・メッセージの前にはエラー番号が付けられています。

Message number
Message text

コネクターは、以下の各セクションで説明するような特定のエラーを処理します。

アプリケーションのタイムアウト

以下のような場合には、エラー・メッセージ ABON_APPRESPONSETIMEOUT が戻されます。

- メッセージ検索中に、コネクターが JMS サービス・プロバイダーとの接続を確立できない。
- コネクターはビジネス・オブジェクトを正常にメッセージに変換したが、接続が切断されたためにメッセージを出力キューに配信できない。
- コネクターはメッセージを発行したが、変換プロパティー TimeoutFatal の値が True であるビジネス・オブジェクトの応答待ちがタイムアウトになった。
- コネクターが戻りコード APP_RESPONSE_TIMEOUT または UNABLE_TO_LOGIN を含む応答メッセージを受信した。

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するメッセージを検索する場合、あるいは gotApplEvent() メソッドから NO_SUBSCRIPTION_FOUND が戻された場合、コネクターは、UnsubscribedDestination プロパティーに指定されたキューにメッセージを配信します。

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

アクティブでないコネクター

gotApplEvent() メソッドが CONNECTOR_NOT_ACTIVE コードを戻すと、pollForEvents() メソッドは APP_RESPONSE_TIMEOUT コードを戻し、イベントは InProgress Destination に置かれたままになります (指定されている場合)。

データ・ハンドラーの変換

データ・ハンドラーがメッセージをビジネス・オブジェクトに変換できなかった場合や (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、ErrorDestination で指定されたキューに送信されます。ErrorDestination が定義されていない場合、エラーが原因で処理できないメッセージは廃棄されます。

データ・ハンドラーがビジネス・オブジェクトの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・メッセージはアダプターにハードコーディングされています。トレースはオプションのデバッグ機能であり、この機能をオンにするとコネクターの動作を密着して追跡できます。トレース・メッセージは、デフォルトでは 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、コネクター構成プロパティーを参照してください。

以下に、コネクターのトレース・メッセージの推奨レベルを示します。

レベル 0　このレベルは、コネクターのバージョンを示すトレース・メッセージに使用します。

レベル 1　このレベルは、処理される各ビジネス・オブジェクトに関するキー情報を提供したり、ポーリング・スレッドが入力キューに新規のメッセージを検出するたびに記録したりするトレース・メッセージに使用します。

レベル 2　ビジネス・オブジェクトが gotApplEvent() または executeCollaboration() からブローカーに送付されるたびに記録されるトレース・メッセージに使用します。

レベル 3　このレベルは、メッセージからビジネス・オブジェクトおよびその反対の変換についての情報を提供したり、メッセージの出力キューへのデリバリーについての情報を提供したりするトレース・メッセージに使用します。

レベル 4　このレベルは、コネクターがある関数を入力または出力する場合を示すトレース・メッセージに使用します。

レベル 5　このレベルは、コネクターの初期化の通知、アプリケーション内で実行されるステートメントの表現、メッセージがキューから出し入れされる際の通知、またはビジネス・オブジェクト・ダンプの記録をするトレース・メッセージに使用します。

開始に関する問題の修正

問題	可能性のある解決方法/説明
コネクターが初期設定中に不意にシャットダウンして、「Exception in thread "main" java.lang.NoClassDefFoundError: javax/jms/JMSException…」というメッセージがレポートされた。	コネクターがファイル jms.jar を検出できません。
コネクターが初期設定中に不意にシャットダウンして、「Exception in thread "main" java.lang.NoClassDefFoundError: javax/naming/Referenceable...」というメッセージがレポートされた。	コネクターがファイル jndi.jar を検出できません。

付録 A. コネクターの標準構成プロパティー

この付録では、WebSphere InterChange Server Express で動作する、WebSphere Business Integration Server Express のアダプターに含まれるコネクター・コンポーネントの標準構成プロパティーについて説明します。

コネクターによっては、一部の標準プロパティーが使用されないことがあります。Connector Configurator Express から統合ブローカーを選択すると、ご使用のアダプターに対して構成する必要がある標準プロパティーのリストが表示されます。

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

標準コネクター・プロパティーの構成

アダプター・コネクターには 2 つのタイプの構成プロパティーがあります。

- 標準構成プロパティー
- コネクター固有のプロパティー

このセクションでは、標準構成プロパティーについて説明します。コネクター固有の構成プロパティーについては、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 Express の使用

コネクター・プロパティーの構成は Connector Configurator Express から行います。Connector Configurator Express には、System Manager からアクセスします。Connector Configurator Express の使用方法の詳細については、付録の『Connector Configurator Express』を参照してください。

プロパティー値の設定と更新

プロパティー・フィールドのデフォルトの長さは 255 文字です。

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

1. デフォルト
2. リポジトリー
3. ローカル構成ファイル
4. コマンド行

コネクターは、始動時に構成値を取得します。実行時セッション中に 1 つ以上のコネクター・プロパティーの値を変更する場合は、プロパティーの**更新メソッド**によって、変更を有効にする方法が決定されます。標準コネクター・プロパティーには、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。
- **コンポーネント再始動**
  System Manager でコネクターを停止してから再始動しなければ、変更が有効になりません。アプリケーション固有コンポーネントまたは統合ブローカーを停止、再始動する必要はありません。
- **サーバー再始動**
  アプリケーション固有のコンポーネントおよび統合ブローカーを停止して再始動しなければ、変更が有効になりません。
- **エージェント再始動**
  アプリケーション固有のコンポーネントを停止して再始動しなければ、変更が有効になりません。

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

標準プロパティーの要約

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

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

表 18. 標準構成プロパティーの要約

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
AdminInQueue	有効な JMS キュー名	*CONNECTORNAME* /ADMININQUEUE	コンポーネント再始動	Delivery Transport は JMS
AdminOutQueue	有効な JMS キュー名	*CONNECTORNAME*/ADMINOUTQUEUE	コンポーネント再始動	Delivery Transport は JMS
AgentConnections	1 から 4	1	コンポーネント再始動	Delivery Transport は IDL
AgentTraceLevel	0 から 5	0	動的	
ApplicationName	アプリケーション名	コネクター・アプリケーション名として指定された値	コンポーネント再始動	
BrokerType	ICS	ICS		
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	

表 18. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
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	IDL または JMS	IDL	コンポーネント再始動	
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
EnableOidForFlowMonitoring	true または false	false	コンポーネント再始動	
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	crossworlds.queue.manager	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>

表 18. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
Locale	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR **注:** これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	true または false	false	コンポーネント再始動	
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE>
MessageFileName	パスまたはファイル名	InterchangeSystem.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	*CONNECTORNAME*/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は true でなければならない。
OADAutoRestartAgent	true または false	false	動的	Repository Directory は <REMOTE>
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE>
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE>
PollEndTime	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	
PollFrequency	正整数 (単位: ミリ秒) no (ポーリングを使用不可にする) key (コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力された場合にのみポーリングする)	10000	動的	
PollQuantity	1 から 500	1	エージェント再始動	JMS トランスポートのみ: Container Managed Events を指定
PollStartTime	HH:MM(HH は 0 から 23、MM は 0 から 59)	HH:MM	コンポーネント再始動	
RepositoryDirectory	メタデータ・リポジトリーの場所		エージェント再始動	<REMOTE> に設定する

表 18. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
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
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 JMS キュー名	*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	有効な JMS キュー名	*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	有効な JMS キュー名	*CONNECTORNAME*/SYNCHRONOUSRESPONSEQUEUE	コンポーネント再始動	Delivery Transport は JMS
WireFormat	CwBO	CwBO	エージェント再始動	

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

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

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

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

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

AgentTraceLevel

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

ApplicationName

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

BrokerType

使用する統合ブローカーを指定します。ICS を指定する必要があります。

CharacterEncoding

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

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

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

ConcurrentEventTriggeredFlows

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

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

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

- Maximum number of concurrent events プロパティーの値を増加して、コラボレーションが複数のスレッドを使用できるように構成します。
- 宛先アプリケーションのアプリケーション固有コンポーネントが複数の要求を並行して実行できることを確認します。つまり、このコンポーネントがマルチスレッド化されているか、またはコネクター・エージェント並列処理を使用でき、複数プロセスに対応するよう構成されている必要があります。Parallel Process Degree 構成プロパティーに、1 より大きい値を設定します。

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

ContainerManagedEvents

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

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

デフォルト値は No value です。

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

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

また、MimeType、DHClass、および DataHandlerConfigMOName (オプショナル) プロパティーを設定したデータ・ハンドラーも構成する必要があります。これらのプロパティーの値を設定するには、Connector Configurator Express 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。

ControllerStoreAndForwardMode

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

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

ます。アプリケーション固有のコンポーネントが作動可能になると、コネクター・コントローラーはアプリケーション固有のコンポーネントにその要求を転送します。

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

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

デフォルト値は `true` です。

ControllerTraceLevel

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

DeliveryQueue

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

コネクターから WebSphere InterChange Server Express 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、`IDL` (CORBA IIOP) または `JMS` (Java Messaging Service) です。デフォルトは IDL です。

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

JMS

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

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

重要: WebSphere InterChange Server Express で動作しているコネクターで JMS トランスポート機構を使用すると、メモリー制限が発生することがあります。

この環境では、WebSphere MQ クライアント内でメモリーが使用されるため、(サーバー側の) コネクター・コントローラーと (クライアント側の) コネクターの両方を始動するのは困難な場合があります。

DuplicateEventElimination

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

このプロパティーは、false に設定することもできます。

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

EnableOidForFlowMonitoring

このプロパティーを true に設定すると、アダプター・フレームワークは、フロー・モニターを使用できるようにするため、着信 **ObjectEventId** を外部キーとしてマークします。

デフォルト値は false です。

FaultQueue

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

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

エージェントの最大ヒープ・サイズ (メガバイト単位)。

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)。

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)。

デフォルト値は 1M です。

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

デフォルト値は crossworlds.queue.manager です。

jms.NumConcurrentRequests

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

デフォルト値は 10 です。

jms.Password

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

デフォルトはありません。

jms.UserName

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

デフォルトはありません。

Locale

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

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、以下のように説明されます。

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

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

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

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

LogAtInterchangeEnd

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

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

MaxEventCapacity

コントローラー・バッファー内のイベントの最大数。このプロパティーは、フロー制御で使用されます。

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

MessageFileName

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

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

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

MonitorQueue

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

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

コネクターが自動再始動およびリモート再始動機能を使用するかどうかを指定します。この機能では、MQ により起動される Object Activation Daemon (OAD) を使用

して、異常シャットダウン後にコネクターを再始動したり、System Monitor からリモート・コネクターを始動したりします。

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

デフォルト値は false です。

OADMaxNumRetry

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

デフォルト値は 1000 です。

OADRetryTimeInterval

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

デフォルト値は 10 です。

PollEndTime

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

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

PollFrequency

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

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

デフォルト値は 10000 です。

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

PollQuantity

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

PollStartTime

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

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

RequestQueue

WebSphere InterChange Server Express からコネクターへビジネス・オブジェクトが送信されるときに使用されるキューです。

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

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

この値は <REMOTE> に設定する必要があります。これは、コネクターが InterChange Server Express リポジトリーからこの情報を取得するためです。

ResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

JMS 応答キューを指定します。JMS 応答キューは、応答メッセージをコネクター・フレームワークから統合ブローカーへデリバリーします。WebSphere InterChange Server Express は、要求を送信した後、JMS 応答キューで応答メッセージを待機します。

RestartRetryCount

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

デフォルト値は 3 です。

RestartRetryInterval

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

デフォルト値は 1 です。

SourceQueue

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

JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、 59 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。設定値は CwBO です。

付録 B. Connector Configurator Express

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

この付録では、次のトピックについて説明します。

Connector Configurator Express の概要

Connector Configurator Express では、WebSphere InterChange Server Express で使用するアダプターのコネクター・コンポーネントを構成できます。

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

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成する。
- **コネクター構成ファイル**を作成します。インストールするコネクターごとに構成ファイルを 1 つ作成する必要があります。
- 構成ファイル内のプロパティーを設定する。
  場合によっては、コネクター・テンプレートでプロパティーに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、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、ロギングとトレース、およびデータ・ハンドラーに関するパラメーターを指定する必要があります。

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

標準プロパティーは、すべてのコネクターで使用されるので、新規に定義する必要はありません。構成ファイルを作成すると、Connector Configurator Express によって標準プロパティーがそのファイルに挿入されます。ただし、Connector Configurator Express で各標準プロパティーの値を設定する必要があります。

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

ただし**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクター固有プロパティーのテンプレートを作成します。システム内で既にテンプレートが作成されている場合には、作成されているテンプレートを使用します。システム内でまだテンプレートが作成されていない場合には、71 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 Express は、Windows 環境でのみ実行できます。別の環境でコネクターを実行する場合には、Windows で Connector Configurator Express を使用して構成ファイルを変更し、このファイルを別の環境へコピーしてください。

Connector Configurator Express の始動

Connector Configurator Express は、以下の 2 種類のモードで始動し、実行することができます。

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

スタンドアロン・モードでの Configurator Express の実行

Connector Configurator Express をブローカーと連携させずに別個に実行して、コネクター構成ファイルを編集することができます。

これを行うには、以下のステップを実行します。

- **「スタート」>「プログラム」**から、**「IBM WebSphere Business Integration Server Express」>「Toolset Express」>「開発」>「Connector Configurator Express」**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。

Connector Configurator Express を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存する方法が便利です (76 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator Express の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

1. 「System Manager」ウィンドウの「コネクター」フォルダーでいずれかの構成ファイルを選択し、右クリックします。

2. 「標準のプロパティー」タブをクリックし、この構成ファイルに含まれているプロパティーを確認します。

コネクター固有のプロパティー・テンプレートの作成

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

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

- テンプレートの新規作成については、71 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でこのテンプレートを保管します。

新規テンプレートの作成

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

テンプレートは以下のように作成します。

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

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

     「テンプレート名」表示に、現在使用可能なすべてのテンプレートの名前が表示されます。
   - テンプレートに含まれているコネクター固有のプロパティー定義を調べるには、**「テンプレート名」**表示でそのテンプレートの名前を選択します。そのテンプレートに含まれているプロパティー定義のリストが**「テンプレートのプレビュー」**表示に表示されます。テンプレートを作成するときには、ご使用のコネクターに必要なプロパティー定義に類似したプロパティー定義が含まれている既存のテンプレートを使用できます。
3. **「テンプレート名」**表示からテンプレートを選択し、その名前を**「名前の検索」**フィールドに入力し (または**「テンプレート名」**で自分の選択項目を強調表示し)、**「次へ」**をクリックします。

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

一般特性の指定

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

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

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

値の指定

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

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

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

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

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

   値が作成されて、グリッドに表示されると、そのテーブルの表示内から編集できるようになります。テーブルにある既存の値の変更を行うには、その行の行番号

をクリックして行全体を選択します。次に**「値」**フィールドを右マウス・ボタンでクリックし、**「値の編集 (Edit Value)」**をクリックします。

依存関係の設定

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

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

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

   == (等しい)

   != (等しくない)

   > (より大)

   < (より小)

   >= (より大か等しい)

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

新しい構成ファイルを作成

コネクター構成ファイルを作成するには、コネクター固有のテンプレートから作成するか、既存の構成ファイルを変更します。

コネクター固有のテンプレートからの構成ファイルの作成

コネクター固有のテンプレートを作成すると、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・ボックス表示されます。
   - **名前**

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

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

     デフォルトのブローカーは ICS です。この値は変更できません。
   - **コネクター固有プロパティー・テンプレートを選択**

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

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

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

既存ファイルの使用

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

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

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

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

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

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

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

     これまでリポジトリー・ファイルを使用してコネクターを構成していた場合は、このオプションを選択します。リポジトリー・ファイルに複数のコネクター定義が含まれている場合は、ファイルを開くとすべての定義が表示されます。
   - すべてのファイル (*.*)

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

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

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

構成ファイルの完成

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

Connector Configurator Express では、以下のセクションに記載されているプロパティーの値を設定する必要があります。

- 77 ページの『標準コネクター・プロパティーの設定』
- 77 ページの『アプリケーション固有の構成プロパティーの設定』
- 78 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』
- 80 ページの『関連付けられたマップ』
- 82 ページの『トレース/ログ・ファイル値の設定』

注: コネクターが JMS メッセージングを使用するものである場合、データをビジネス・オブジェクトに変換するデータ・ハンドラーを構成できるように、追加のカテゴリーが表示されることがあります。詳細については、 82 ページの『データ・ハンドラー』を参照してください。

構成ファイル・プロパティーの設定

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

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

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

「標準プロパティー」と**「コネクター固有プロパティー」**のフィールドは、どのフィールドが構成可能であるかを示すために色分けされています。

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

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下の手順を実行します。

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

アプリケーション固有の構成プロパティーの設定

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

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

4. 77 ページの『標準コネクター・プロパティーの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

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

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

コネクター・プロパティーの暗号化

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

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

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

更新メソッド

付録『*コネクターの標準構成プロパティー*』の 53 ページの『プロパティー値の設定と更新』にある更新メソッドの説明を参照してください。

コネクター・プロパティーはほとんどが静的なプロパティーであり、それらの**更新メソッド**は**コンポーネント再始動**です。変更を有効にするには、変更したコネクター構成ファイルを保管した後、コネクターを再始動する必要があります。

サポートされるビジネス・オブジェクト定義の指定

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

サポートされるビジネス・オブジェクトを指定するときには、指定するビジネス・オブジェクトとそのオブジェクトに対応するマップが、システムに存在していなければなりません。ビジネス・オブジェクト定義 (データ・ハンドラー・メタオブジェクトのビジネス・オブジェクト定義を含みます) とマップ定義は、統合コンポー

ネント・ライブラリー (ICL) プロジェクトに保管されている必要があります。ICL プロジェクトの詳細については、WebSphere Business Integration Server Express の「*ユーザーズ・ガイド*」を参照してください。

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

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

ビジネス・オブジェクト名

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

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

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

エージェント・サポート

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

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

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

最大トランザクション・レベル

コネクタ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。

関連付けられたマップ

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

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

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

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

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的**

  場合によっては、関連マップを明示的にバインドすることが必要になります。

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

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

  以下の手順を実行して、マップを明示的にバインドします。

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

リソース

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

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

トレース/ログ・ファイル値の設定

コネクター構成ファイルまたはコネクター定義ファイルを開くと、Connector Configurator Express は、そのファイルに含まれるロギングとトレースに関する値をデフォルト値として使用します。これらの値は、Connector Configurator Express 内で変更できます。

ログとトレースの値を変更するには、以下の手順を実行します。

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

     注: STDOUT オプションは、Windows プラットフォームで実行しているコネクターの**「トレース/ログ・ファイル」**タブでのみ使用できます。
   - ファイルに:
     ログ・メッセージまたはトレース・メッセージを指定されたファイルに書き込みます。ファイルを指定するには、ディレクトリー・ボタン (省略符号) をクリックし、指定する格納場所に移動し、ファイル名を指定し、**「保管」**をクリックします。(コネクターが、Connector Configurator Express をインストールした Windows プラットフォームで実行されていない場合は、最初に、システム上のファイルの格納場所にドライブをマップする必要があります。)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。このタブは、アダプターが保証付きイベント・デリバリーを利用するものである場合に使用可能になります。

これらのプロパティーに使用する値については、標準プロパティーに関する付録の『`ContainerManagedEvents`』の説明を参照してください。

構成ファイルの保管

構成ファイルの作成とそのファイルに含まれるプロパティーの設定が完了したら、使用するコネクターに応じた適切な場所にそのファイルを配置する必要があります。ICL プロジェクトに構成を保管し、保管されたファイルを System Manager から InterChange Server Express へロードしてください。

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

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

System Manager でのプロジェクトの使用方法と、配置の詳細については、IBM WebSphere Business Integration Server Express の「*ユーザーズ・ガイド*」を参照してください。

構成の完了

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

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

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

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

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

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

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

例えば、Locale プロパティーの値のリストにロケール en_GB を追加するには、stdConnProps.xml ファイルを開き、以下に太文字で示した行を追加してください。

```
<Property name="Locale"
isRequired="true"
updateMethod="component restart">
                <ValidType>String</ValidType>
            <ValidValues>
                                <Value>ja_JP</Value>
```

```
                                <Value>ko_KR</Value>
                                <Value>zh_CN</Value>
                                <Value>zh_TW</Value>
                                <Value>fr_FR</Value>
                                <Value>de_DE</Value>
                                <Value>it_IT</Value>
                                <Value>es_ES</Value>
                                <Value>pt_BR</Value>
                                <Value>en_US</Value>
                                <Value>en_GB</Value>

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

付録 C. トピック・ベースおよびキュー・ベースのメッセージングの構成

- 『キュー・ベース・メッセージングの構成』
- 86 ページの『トピック・ベース・メッセージングの構成』

この付録では、共通の JMS プロバイダーとして WebSphere MQ を使用して、Adapter for JMS を構成する方法について説明します。詳細については、「*WebSphere MQ Using Java* 」ガイドを参照してください。

注: WebSphere MQ を JMS プロバイダーとして使用する場合、統合用として、WebSphere MQ にアダプターを使用することを強くお勧めします。以下のステップは、共通の JMS プロバイダーを使用して JMS アダプターを構成する方法を示す参照用としてのみ記載されています。

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。

キュー・ベース・メッセージングの構成

1. WebSphere MQ および WebSphere MQ クライアント・ライブラリー (JMS サポートを含む) をインストールする。
2. `fscontext.jar` および `providerutil.jar` を含め、すべての MQ クライアント・ライブラリーが使用しているシステム・クラスパスにあることを確認する。あるいは、`jmsAdmin.bat` ファイルを変更し、`-Djava.ext.dirs="<MQ のホーム・ディレクトリー>/Java/lib` を Java コマンド行スクリプトに追加して、ツールがすべてのクライアント・ライブラリー・ファイルを使用できるようにする。ツールによって報告される ClassDefNotFoundExceptions は、ライブラリーが欠落したことが原因であることに注意します。クラスパスを再チェックしてください。
3. `<MQ のホーム・ディレクトリー>Java/bin/jmsAdmin.config` を開き、次のプロパティーを設定する。
   - `INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory`
   - `PROVIDER_URL=file://c:/temp`
   - `SECURITY_AUTHENTICATION=none`
4. 以下を含む `MyJNDI.txt` という名前のファイルを作成する。
   ```
   DEFINE QCF(MyQCF)
   HOST(<ホスト名>)  +PORT(<1414 などの MQ リスナー・ポート名>)  +
   CHANNEL(<CHANNEL1 などの MQ サーバー接続チャネル名>) +
   QMGR(<MQ キュー・マネージャー名>) +
   TRAN(client)
   END
   ```
5. `<MQ のホーム・ディレクトリー>/java/bin/jmsAdmin.bat < MyJNDI.txt` を実行して、オブジェクトを JNDI 名にバインドする。
6. 次の JMS コネクター固有プロパティーを、以下に示すように構成する。

```
CTX_InitialContextFactory = com.sun.jndi.fscontext.RefFSContextFactory
CTX_ProviderURL = file://c:/temp
ConnectionFactoryName = MyQCF
```

トピック・ベース・メッセージングの構成

1. WebSphere MQ および WebSphere MQ クライアント・ライブラリー (JMS サポートを含む) をインストールする。
2. fscontext.jar および providerutil.jar を含め、すべての MQ クライアント・ライブラリーが使用しているシステム・クラスパスにあることを確認する。あるいは、jmsAdmin.bat ファイルを変更し、-Djava.ext.dirs="*<MQ のホーム・ディレクトリー>*/Java/lib を Java コマンド行スクリプトに追加して、ツールがすべてのクライアント・ライブラリー・ファイルを使用できるようにする。ツールによって報告される ClassDefNotFoundExceptions は、ライブラリーが欠落したことが原因であることに注意します。クラスパスを再チェックしてください。
3. 適切な WebSphere MQ MA0C SupportPac を IBM からダウンロードし、それをインストールして、トピック・ベース (パブリッシュ/サブスクライブ) のメッセージング・サポートを MQ で使用可能にする。例えば、ma0c_ntmq52 を検索すると、Windows では MQ 5.2 用のトピック・ベース・メッセージング・パッチが見つかります。
4. ディレクトリーを *<MQ のホーム・ディレクトリー>*/Java/bin に変更し、runmqsc < MQJMS_PSQ.mqsc を実行する。
5. IVTSetup.bat を実行する。プロセスは、エラーを報告せずに、「Done!」と表示されるはずです。
6. *<MQ* のホーム・ディレクトリー>Java/bin/jmsAdmin.config を開き、次のプロパティーを設定する。
   - INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory
   - PROVIDER_URL=file://c:/temp
   - SECURITY_AUTHENTICATION=none
7. 以下を含む MyJNDI.txt という名前のファイルを作成する。
   ```
   DEFINE TCF(MyTCF) HOST(<ホスト名>)  +PORT(<1414 などの MQ リスナー・ポート名>)  +
   CHANNEL(<CHANNEL1 などの MQ サーバー接続チャネル名>) +
   QMGR(<MQ キュー・マネージャー名>) +
   TRAN(client)
   END
   ```
8. *<MQ のホーム・ディレクトリー>*/java/bin/jmsAdmin.bat < MyJNDI.txt を実行して、オブジェクトを JNDI 名にバインドする。
9. 次の JMS コネクター固有プロパティーを、以下に示すように構成する。

   ```
   CTX_InitialContextFactory = com.sun.jndi.fscontext.RefFSContextFactory
   CTX_ProviderURL = file://c:/temp
   ConnectionFactoryName = MyTCF
   ```

特記事項

特記事項

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

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

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

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

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

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

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

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

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

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

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

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

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

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

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

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

著作権使用許諾

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを

経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

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。

汎用プログラミング・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。

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

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

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

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

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

WebSphere Business Integration Server Express V4.3.1 および WebSphere Business Integration Server Express Plus V4.3.1

Printed in Japan

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