WebSphere Business Integration Server
Express and Express Plus

Adapter for WebSphere MQ 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、 95 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 および IBM WebSphere Business Integration Server Express Plus バージョン 4.3.1 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
Adapter for WebSphere MQ 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

目次

本書について

製品 IBM(R)WebSphere Business Integration Server Express および IBM(R) WebSphere Business Integration Server Express Plus 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。

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

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

対象読者

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

本書の前提条件

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

関連文書

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

本書では、以下の規則を使用します。

`Courier フォント`	コマンド名、ファイル名、ユーザーの入力した情報、システムが画面に出力した情報など、記述されたとおりの値を示します。
太字	初出語を示します。
イタリック、***イタリック***	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示します。
`< >`	命名規則により、1 つの名前の各エレメントを個々に判別できるようにするために、不等号括弧で囲みます。例えば、`<server_name><`*`connector_name`*`>tmp.log` のように使用します。
/、¥	本書では、ディレクトリー・パスに円記号 (¥) を使用します。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 製品のパス名は、ご使用のシステムでの製品インストール・ディレクトリーからの相対パスです。
`%text%` と *`$text`*	% 記号で囲まれたテキストは、Windows の `text` システム変数またはユーザー変数の値を示します。UNIX 環境の場合、これに相当する表記は *`$text`* になります。これは、*`text`* UNIX 環境変数の値を示します。
ProductDir	製品のインストール先ディレクトリーを表します。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、以下のプラットフォームのサポートが追加されました。

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

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 概要

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

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

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

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

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

注: すべての WebSphere Business Integration Server Express アダプターは、InterChange Server Express Integration Broker と動作します。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。

コネクター・アーキテクチャー

コネクターはメタデータ主導型です。メッセージのルーティングおよびフォーマット変換は、イベント・ポーリング技法によって開始されます。コネクターは、Java™ Message Service (JMS) の MQ インプリメンテーションを使用します。JMS は、エンタープライズ・メッセージング・システムにアクセスするための API で、保証付きイベント・デリバリーも可能になります。

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

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

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

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

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

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

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

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

アプリケーションとコネクター間の通信方法

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

メッセージ要求

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

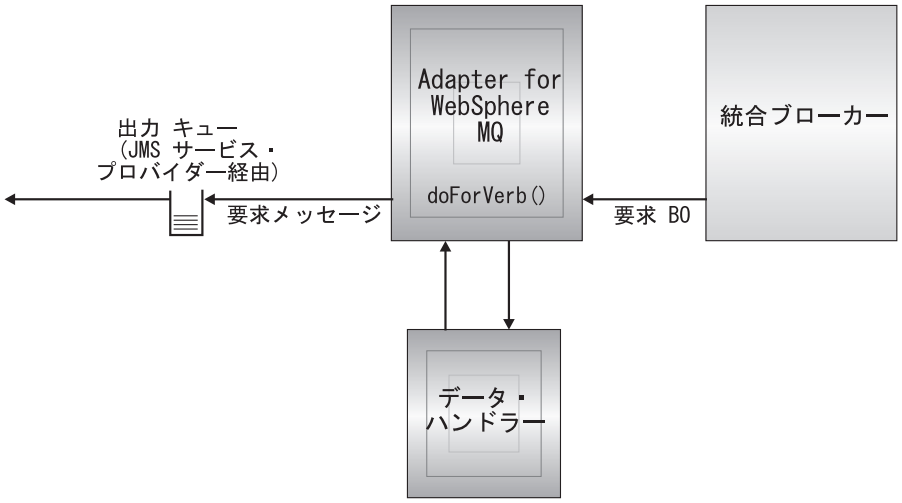

図 1. アプリケーションとコネクターの間の通信方法: メッセージ要求

イベント・デリバリー

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

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

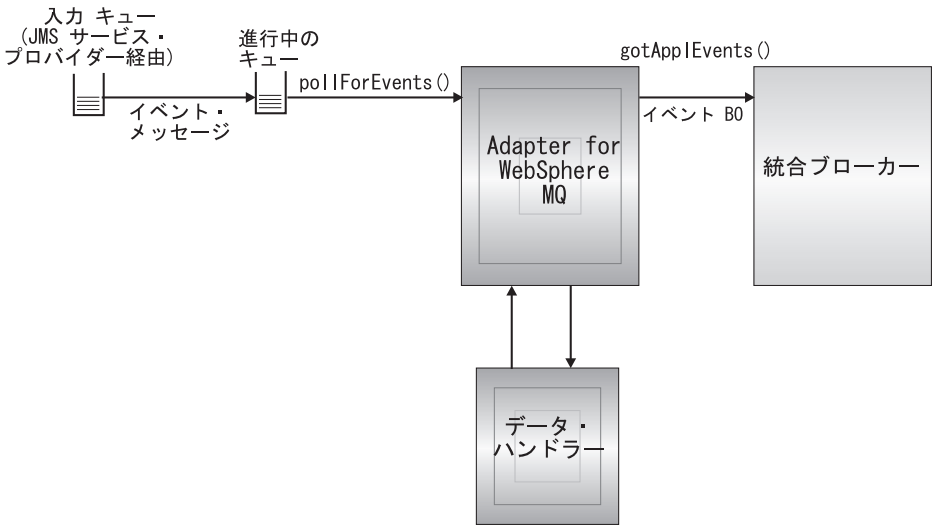

図 2. アプリケーションとコネクターの間の通信方法: イベント・デリバリー

イベント処理

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

検索

コネクターは、pollForEvents() メソッドを使用して MQ キューからメッセージを定期的にポーリングします。メッセージを検出すると、コネクターはそれを MQ キューから検索して調べ、メッセージのフォーマットを判別します。判別されたフォーマットがコネクターの静的オブジェクトで定義されている場合、コネクターは、メッセージの本体とそのフォーマットに関連付けられているビジネス・オブジェクトの新しいインスタンスの両方を、構成されているデータ・ハンドラーに渡します。データ・ハンドラーはビジネス・オブジェクトを取り込み、動詞を指定すると想定されています。判別されたフォーマットが静的メタオブジェクトで定義されていない場合、コネクターはメッセージの本体のみをデータ・ハンドラーに渡します。データ・ハンドラーはメッセージに対する正しいビジネス・オブジェクトを判別して作成し、取り込むと想定されています。イベント失敗のシナリオについては、58 ページの『エラー処理』を参照してください。

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

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

同期イベント処理

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

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

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

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

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

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

MQMD フィールド	説明	サポートされる値 (複数の値をサポートするには論理和を使用)
report	要求されたレポート・メッセージのオプション	次のいずれか一方、または両方を指定できます。 • MQRO_PAN コネクターは、ビジネス・オブジェクトが正常に処理された場合にレポート・メッセージを送信します。 • MQRO_NAN コネクターは、ビジネス・オブジェクトの処理中にエラーが発生した場合にレポート・メッセージを送信します。 次のいずれかの値を指定すると、レポート・メッセージの相関 ID の設定方法を制御できます。 • MQRO_COPY_MSG_ID_TO_CORREL_ID コネクターは、要求メッセージのメッセージ ID をレポートの相関 ID にコピーします。これはデフォルトのアクションです。 • MQRO_PASS_CORREL_ID コネクターは、要求メッセージの相関 ID をレポートの相関 ID にコピーします。
ReplyToQueue	応答キューの名前	レポート・メッセージの送信先となるキューの名前。
replyToQueueManager	キュー・マネージャーの名前	レポート・メッセージの送信先となるキュー・マネージャーの名前。
メッセージの本体		コネクターに構成されているデータ・ハンドラーと互換性のあるフォーマットで直列化されたビジネス・オブジェクト。

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

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

次の表に、コネクターから要求発行者に返送されるレポートの構造を示します。

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

MQMD フィールド	説明	サポートされる値 (複数の値がある場合は OR として扱います)
feedback	レポートのタイプ	次のいずれかです。 • MQRO_PAN コラボレーションがビジネス・オブジェクトを正常に処理した場合に、レポートが返送されます。 • MQRO_NAN要求の処理中にコネクターまたはコラボレーションがエラーを検出した場合に、レポートが返送されます。
メッセージの本体		コラボレーションがビジネス・オブジェクトを正常に処理した場合、コネクターはメッセージの本体にコラボレーションから戻されたビジネス・オブジェクトを取り込みます。このデフォルトの動作は、静的メタデータ・オブジェクトの DoNotReportBusObj プロパティーを true に設定することによりオーバーライドできます。 要求を処理できなかった場合、コネクターはメッセージの本体にコネクターまたはコラボレーションによって生成されたエラー・メッセージを取り込みます。

リカバリー

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

始動時の失敗 (Fail on startup)

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

再処理 (Reprocess)

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

無視 (Ignore)

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

エラー・ログ記録 (Log error)

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

アーカイブ

コネクターのプロパティー ArchiveQueue が指定されており、かつ有効なキューを示している場合には、コネクターは正常に処理されたすべてのメッセージのコピーをアーカイブ・キューに格納します。ArchiveQueue が未定義の場合、メッセージは処理後に破棄されます。アンサブスクライブされたメッセージまたはエラーを含むメッセージのアーカイブの詳細については、55 ページの『第 3 章 ビジネス・オブジェクトの作成および変更』の 58 ページの『エラー処理』を参照してください。

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

保証付きイベント・デリバリー

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

注: 保証付きイベント・デリバリー機能を使用しないと、コネクターがイベントをパブリッシュして (コネクターが pollForEvents() メソッド内の gotApplEvent() メソッドを呼び出して) から、イベント・レコードを削除してイベント・ストアを更新する (または「イベント通知済み」状況に更新する) までの間に、障害の可能性を示す短い間が空きます。このすき間で障害が発生すると、イベントは送信されますが、イベント・レコードはイベント・ストアで「進行中」状況のままになっています。コネクターは再始動時に、このイベント・ストアに残されたイベント・レコードを検出して送信するので、イベントが 2 回送信されることになります。

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

リバリーを行うようにコネクターを構成するには、「システム・インプリメンテーション・ガイド」の説明を参照してください。

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

ビジネス・オブジェクト要求

ビジネス・オブジェクト要求は、InterChange Server Express が `doVerbFor()` メソッドにビジネス・オブジェクトを送信するときに処理されます。コネクターは、構成されているデータ・ハンドラーを使用してビジネス・オブジェクトを WebSphere MQ メッセージに変換し、発行します。データ・ハンドラーについての要件を除いては、処理されるビジネス・オブジェクトのタイプに関する要件はありません。

動詞の処理

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

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

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

Create、Update、および Delete

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

非同期デリバリー

これは、Create 動詞、Update 動詞、および Delete 動詞を持つビジネス・オブジェクトのデフォルト・デリバリー・モードです。データ・ハンドラーを使用して、ビジネス・オブジェクトからメッセージが作成され、出力キューに書き込まれます。

メッセージがデリバリーされた場合、コネクターは BON_SUCCESS を戻します。それ以外の場合は、BON_FAIL を戻します。

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

同期デリバリー

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

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

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

* は、IBM によって定義される定数を示します。

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

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

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

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

フィールド	説明	値
MessageType	メッセージ・タイプ	MQMT_REPORT*

* は、IBM によって定義される定数を示します。

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

WebSphere MQ フィードバック・コード	対応する WebSphere Business Integration Server Express の応答*
MQFB_PAN または MQFB_APPL_FIRST	SUCCESS
MQFB_NAN または MQFB_APPL_FIRST + 1	FAIL
MQFB_APPL_FIRST + 2	VALCHANGE
MQFB_APPL_FIRST + 3	VALDUPES
MQFB_APPL_FIRST + 4	MULTIPLE_HITS
MQFB_APPL_FIRST + 5	FAIL_RETRIEVE_BY_CONTENT
MQFB_APPL_FIRST + 6	BO_DOES_NOT_EXIST
MQFB_APPL_FIRST + 7	UNABLE_TO_LOGIN
MQFB_APPL_FIRST + 8	APP_RESPONSE_TIMEOUT (この応答後、コネクター・エージェントは即時に終了します)

* 詳細については、「*システム・インプリメンテーション・ガイド*」を参照してください。

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

コネクターは、応答メッセージを検索すると、デフォルトでは、応答の correlationID を要求メッセージの messageID と突き合わせます。続いて、要求を発行したスレッドに通知を送信します。コネクターは、応答の feedback フィールドの設定によって、メッセージの本体にビジネス・オブジェクトとエラー・メッセージのどちらが含まれているかを予測します。ビジネス・オブジェクトが含まれてい

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

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

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

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

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

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

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

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

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

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

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

これで、応答メッセージが識別されます。

また、次のように、複数の置換を指定することも可能です。

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

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

次のいずれかの状況が発生すると、実行時にエラーが報告されます。

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

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

コネクターはこのメッセージ選択子を次のように解決します。

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

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

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

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

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

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

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

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

Retrieve、Exists、および Retrieve By Content

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

次の表に、これらの動詞に対応する応答メッセージを示します。

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

ロケール依存データの処理

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、ほとんどの既知の文字コード・セット (1 バイト系とマルチバイト系を含む) の文字に対応できるエンコード方式が組み込まれています。WebSphere Business Integration システムのほとんどのコンポーネントは Java で記述されています。したがって、ほとんどの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、63 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。

共通の構成タスク

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

アダプターのインストール

何をどこにインストールするかについての詳細は、19 ページの『第 2 章 アダプターのインストールおよび構成』を参照してください。

コネクター・プロパティーの構成

コネクターには、標準構成プロパティーとコネクター固有の構成プロパティーの 2 種類の構成プロパティーがあります。一部のプロパティーはデフォルト値を持っており、変更を加えなくても使用できます。また、一部のプロパティーについては、コネクターを実行する前に値を設定する必要があります。詳細については、19 ページの『第 2 章 アダプターのインストールおよび構成』を参照してください。

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

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

通知なしで要求を送信するためのコネクターの構成

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

- 送信する要求を表しており、コネクター向けに構成したデータ・ハンドラーとの互換性もあるビジネス・オブジェクトを作成します。
- 静的メタオブジェクトまたは動的メタオブジェクトを使用して、宛先のキューとフォーマットを指定します。静的メタオブジェクトと動的メタオブジェクトの詳細については、33 ページの『静的メタオブジェクト』と 40 ページの『動的な子メタオブジェクト』を参照してください。
- (静的または動的) メタオブジェクト内のプロパティー ResponseTimeout を -1 に設定します。この設定では、コネクターは発行したビジネス・オブジェクトの戻りをチェックしません。
- 詳細については、9 ページの『Create、Update、および Delete』、33 ページの『メタオブジェクト属性構成』、および 55 ページの『第 3 章 ビジネス・オブジェクトの作成および変更』を参照してください。

要求を送信して通知を取得するためのコネクターの構成

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

- コネクターが応答を待機する時間を指示するために正の ResponseTimeout 値を指定する点を除いて、 15 ページの『通知なしで要求を送信するためのコネクターの構成』の説明にある手順に従います。
- コネクターが予期する応答メッセージの具体的な詳細については、9 ページの『Create、Update、および Delete』を参照してください。示されている要件を応答メッセージが満たしていない場合、コネクターはエラーを報告したり、応答メッセージを認識できなかったりする可能性があります。33 ページの『メタオブジェクト属性構成』と 55 ページの『第 3 章 ビジネス・オブジェクトの作成および変更』も参照してください。

静的メタオブジェクトの構成

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

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

- 33 ページの『静的メタオブジェクト』の手順に従います。
- コネクター固有のプロパティー ConfigurationMetaObject 内で静的メタオブジェクトの名前を指定することにより、コネクターが静的メタオブジェクトにサブスクライブするようにします。詳細については、23 ページの『コネクター固有のプロパティー』を参照してください。

動的メタオブジェクトの構成

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

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

コネクターは動的メタオブジェクトをチェックし、その情報を使用してビジネス・オブジェクトの処理方法を判別します。詳細については、40 ページの『動的な子メタオブジェクト』を参照してください。

MQMD フォーマットの構成

MQMD はメッセージ記述子です。MQMD には、メッセージがアプリケーション間で送信されるときにアプリケーション・データに添付される制御情報が格納されます。静的メタオブジェクトまたは動的メタオブジェクト内で、MQMD 属性 OutputFormat の値を指定する必要があります。詳細については、9 ページの『Create、Update、および Delete』を参照してください。

キュー URI の構成

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

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

  queue://<キュー・マネージャー名>/<実際のキュー>
- コネクター固有の構成プロパティーに、キュー・マネージャーのホストを指定します。
- ターゲット・アプリケーションが MQMD ヘッダーのみを予期していて、JMS クライアントが使用する拡張 MQRFH2 ヘッダーを処理できない場合は、?targetClient=1 をキュー URI に付加します。詳細については、32 ページの『キューの Uniform Resource Identifiers (URI)』と WebSphere MQ のプログラミング・ガイドを参照してください。

データ・ハンドラーの構成

データ・ハンドラーを構成する方法は 2 つあります。

- コネクター固有のプロパティー DataHandlerClassName 内で、データ・ハンドラー・クラス名を指定します。詳細については、23 ページの『コネクター固有のプロパティー』を参照してください。
- または、コネクター固有のプロパティー DataHandlerMimeType および DataHandlerConfigMO 内で、MIME タイプとその MIME タイプの構成を定義するデータ・ハンドラー・メタオブジェクトをそれぞれ指定します。詳細については、「*データ・ハンドラー・ガイド*」を参照してください。

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。

前提条件

前提条件ソフトウェア

- コネクターは、WebSphere MQ 5.1、5.2、および 5.3 を介してアプリケーションとのインターオペラビリティーをサポートします。したがって、次のいずれかのソフトウェア・リリースをインストールしている必要があります。

  注: このアダプターは、WebSphere MQ 5.3 環境で SSL (Secure Socket Layer) をサポートしていません。アダプター・フレームワークと統合ブローカーの通信にとって適切な WebSphere MQ ソフトウェア・バージョンについては、ご使用のプラットフォーム用の「WebSphere Business Integration Server Express インストール・ガイド」を参照してください。
- さらに、IBM WebSphere MQ Java クライアント・ライブラリーも必要です。
- コネクターは、以下のプラットフォーム上で実行されます。
  - Microsoft Windows 2000
  - Microsoft Windows 2003
  - OS/400 V5R2、V5R3
  - Red Hat Enterprise Linux AS 3.0
  - SuSE Linux Enterprise Server 8.1 (SP3 を適用)

インストール作業の概要

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

- **統合ブローカーのインストール** この作業では、WebSphere Business Integration Server Express をインストールおよび始動します。ご使用のプラットフォーム用の「WebSphere Business Integration Server Express インストール・ガイド」を参照してください。

- **アダプターおよび関連ファイルのインストール** この作業では、アダプターのファイルをソフトウェア・パッケージから使用システムにインストールします。『アダプターおよび関連ファイルのインストール』を参照してください。

アダプターおよび関連ファイルのインストール

アダプターのインストールの詳細については、Windows、Linux、または OS/400 用の「WebSphere Business Integration Server Express インストール・ガイド」を参照してください。このガイドは、以下のサイトの WebSphere Business Integration Server Express Infocenter にあります。

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、インストール後の製品のパスとファイル名について説明します。

注: 通常、WebSphere MQ と JMS は異なるディレクトリーにインストールされます。

Windows の場合、通常、WebSphere MQ は ¥Program Files¥IBM¥WebSphere MQ の下にインストールされ、JMS は ¥Program Files¥IBM¥WebSphere MQ¥Java¥lib の下にインストールされます。これに従って、WebSphere MQ コネクター始動スクリプト内のクラスパスを変更してください。

Linux の場合、WebSphere MQ は /opt/mqm の下にインストールされ、JMS サポートは /opt/mqm/java/lib の下にインストールされます。

OS400 の場合、WebSphere MQ は /QIBM/ProdData/mqm ディレクトリーにインストールされます。JMS サポートは /QIBM/ProdData/mqm/java/lib にインストールされます。

Windows 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により、コネクターが *ProductDir*¥connectors¥WebSphereMQ ディレクトリーにインストールされます。*ProductDir* は、WebSphere Business Integration Server Express のインストール・ディレクトリーを表す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。

21 ページの表 1 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 1. WebSphere MQ コネクターのインストール済みファイルの構造 (Windows)

***ProductDir* のサブディレクトリー**	**説明**
connectors¥WebSphereMQ	以下のファイルが格納されています。 • CWWebSphereMQ.jar—WebSphere MQ コネクターが使用するクラス • start_WebSphereMQ.bat—コネクターの始動スクリプト • start_WebSphereMQ_service.bat—コネクターを Windows のサービスとして開始するための始動スクリプト。 • ReadMe.htm—アダプターのパッチ情報。
connectors¥messages	WebSphereMQConnector.txt メッセージ・ファイルおよび WebSphereMQConnector_*ll*_*TT*.txt ファイル (言語 (*ll*) および国または地域 (*TT*) に固有のメッセージ・ファイル) が格納されています。
repository¥WebSphereMQ	コネクター用のリポジトリー定義 CN_WebSphereMQ.txt が格納されています。
connectors¥WebSphereMQ¥samples	サンプルの成果物が格納されています。
lib	WBIA.jar ファイルが格納されています。
bin	CWConnEnv.bat ファイルが含まれています。

インストーラーによって、コネクターのアイコンが「スタート」メニューに追加されます (**「スタート」 ->「プログラム」->「IBM WebSphere Business Integration Express」->「Adapters」->「Connectors」** をクリックすることにより使用可能)。コネクターをすばやく始動するには、このファイルへのショートカットをデスクトップに作成してください。

OS/400 のファイル構造

デフォルトでは、アダプターは /QIBM/ProdData/WBIServer43/product ディレクトリー (*ProductDir* と呼ぶ) にインストールされます。表 2 に、コネクターが使用する OS/400 ファイル構造と、標準インストール時にインストールされるファイルを示します。

表 2. WebSphere MQ コネクターのインストール済みファイルの構造 (OS/400)

ProductDir **のサブディレクトリー**	**説明**
connectors/WebSphereMQ	CWWebSphereMQ.jar ファイルおよび start_WebSphereMQ.sh 始動スクリプトが格納されています。
connectors/messages	WebSphereMQConnector.txt メッセージ・ファイルおよび WebSphereMQConnector_*ll*_*TT*.txt ファイル (言語 (*ll*) および国または地域 (*TT*) に固有のメッセージ・ファイル) が格納されています。

表 2. *WebSphere MQ* コネクターのインストール済みファイルの構造 *(OS/400) (続き)*

ProductDir のサブディレクトリー	説明
repository/WebSphereMQ	コネクター用のリポジトリー定義 CN_WebSphereMQ.txt が格納されています。
connectors/WebSphereMQ/samples	サンプルの成果物が格納されています。
lib	WBIA.jar ファイルが格納されています。
bin	CWConnEnv.sh ファイルが格納されています。

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

Linux のファイル構造

デフォルトでは、コネクターに関連付けられている標準ファイルは、*ProductDir*/connectors/WebSphereMQConnector ディレクトリーにインストールされます。*ProductDir* は、WebSphere Business Integration Server Express のインストール・ディレクトリー (デフォルトでは /home/${username}/IBM/WebSphereServer) を表すことに注意してください。

表 3 に、コネクターが使用する Linux ファイル構造と、標準インストール時に自動的にインストールされるファイルを示します。

表 3. *WebSphere MQ* コネクターのインストール済みファイルの構造 *(Linux)*

ProductDir のサブディレクトリー	説明
connectors/WebSphereMQ	CWWebSphereMQ.jar ファイルおよび start_WebSphereMQ.sh 始動スクリプトが格納されています。 start_WebSphereMQ.sh 始動スクリプトは、汎用コネクター・マネージャー・スクリプト (*ProductDir*/bin ディレクトリー内の connector_manager -start スクリプト) から呼び出されます。
connectors/messages	WebSphereMQConnector.txt メッセージ・ファイルおよび WebSphereMQConnector_*ll_TT*.txt ファイル (言語 (*ll*) および国または地域 (*TT*) に固有のメッセージ・ファイル) が格納されています。
repository/WebSphereMQ	コネクター用のリポジトリー定義 CN_WebSphereMQ.txt が格納されています。
connectors/WebSphereMQ/samples	サンプルの成果物が格納されています。
lib	WBIA.jar ファイルが格納されています。
bin	CWConnEnv.sh ファイルが格納されています。

Linux システムでコネクターを始動するには、connector_manager コマンドを使用する必要があります。

コネクター構成

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

コネクターのプロパティーを構成するには、Connector Configurator 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、79 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』、および 63 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、『コネクター固有のプロパティー』を参照してください。

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

標準コネクター・プロパティー

標準構成プロパティーにより、すべてのコネクターによって使用される情報が提供されます。標準構成プロパティーの資料については、63 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

注: Connector Configurator Express で構成プロパティーを設定するときは、BrokerType プロパティーを InterChange Server Express に設定します。InterChange Server Express に関連するプロパティーは、「Connector Configurator Express」ウィンドウに表示されます。

コネクター固有のプロパティー

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

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

名前	指定可能な値	デフォルト値	必須
ApplicationPassword	*ログイン・パスワード*		いいえ
ApplicationUserName	*ログイン・ユーザー ID*		いいえ
ArchiveQueue	*正常に処理されたメッセージのコピーが送信されるキュー*	queue://crossworlds.queue.manager/MQCONN.ARCHIVE	いいえ
CCSID	*キュー・マネージャーの接続に使用する文字セット*	null	いいえ
Channel	*MQ サーバー・コネクター・チャネル*		はい
ConfigurationMetaObject	*構成メタオブジェクトの名前*		はい
DataHandlerClassName	*データ・ハンドラー・クラス名*	com.crossworlds.DataHandlers.text.xml	いいえ
DataHandlerConfigMO	*データ・ハンドラー・メタオブジェクト*	MO_DataHandler_ Default	はい
DataHandlerMimeType	*MIME ファイル・タイプ*	text/xml	いいえ
DefaultVerb	*コネクターがサポートする任意の動詞*	Create	
ErrorQueue	*未処理のメッセージ・キュー*	queue://crossworlds.queue.manager/MQCONN.ERROR	いいえ
FeedbackCodeMappingMO	*フィードバック・コード・メタオブジェクト*		いいえ
HostName	*WebSphere MQ サーバー*		はい
InDoubtEvents	FailOnStartup Reprocess IgnoreLogError	Reprocess	いいえ
InputQueue	*ポーリング・キュー*	queue://crossworlds.queue.manager/MQCONN.IN	いいえ
InProgressQueue	*進行中のイベント・キュー*	queue://crossworlds.queue.manager/MQCONN.IN_PROGRESS	いいえ
PollQuantity	*InputQueue プロパティー内で指定された各キューから検索されるメッセージの数*	1	いいえ
Port	*WebSphere MQ リスナー用に設定されたポート*		はい
ReplyToQueue	*コネクターからの要求発行時に応答メッセージが配信されるキュー*	queue://crossworlds.queue.manager/MQCONN.REPLYTO	いいえ
UnsubscribedQueue	*アンサブスクライブされたメッセージが送信されるキュー*	queue://crossworlds.queue.manager/MQCONN.UNSUBSCRIBE	いいえ
UseDefaults	true または false	false	

ApplicationPassword

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

デフォルト = 設定値なし

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

ApplicationUserName

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

デフォルト = 設定値なし

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

ArchiveQueue

正常に処理されたメッセージのコピーが送信されるキューです。

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

CCSID

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

デフォルト値はヌルです。

Channel

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

デフォルト = 設定値なし

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

ConfigurationMetaObject

コネクターの構成情報を含む静的なメタオブジェクトの名前です。

デフォルト = 設定値なし

DataHandlerClassName

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

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

DataHandlerConfigMO

構成情報を提供するために、データ・ハンドラーに渡されるメタオブジェクト。

デフォルト = MO_DataHandler_Default

DataHandlerMimeType

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

デフォルト = text/xml

DefaultVerb

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

デフォルト= Create

ErrorQueue

処理されなかったメッセージが送信されるキューです。

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

FeedbackCodeMappingMO

メッセージの受信を InterChange Server Express に同期的に確認するために使用されるデフォルトのフィードバック・コードをオーバーライドして再割り当てするプロパティーです。このプロパティーを使用すると、フィードバック・コードを表示するために各属性名を解釈するときのメタオブジェクトを指定できます。フィードバック・コードの対応する値は、InterChange Server Express に渡される戻り状況値です。デフォルトのフィードバック・コードのリストは、10 ページの『同期デリバリー』を参照してください。コネクターは、WebSphere MQ 固有のフィードバック・コードを表す以下の属性値を受け入れます。

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

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

- SUCCESS
- FAIL
- APP_RESPONSE_TIMEOUT
- MULTIPLE_HITS
- UNABLE_TO_LOGIN
- VALCHANGE
- VALDUPES

以下の表に、サンプル・メタオブジェクトを示します。

属性名	デフォルト値
MQFB_APPL_FIRST	SUCCESS

MQFB_APPL_FIRST + 1	FAIL
MQFB_APPL_FIRST + 2	UNABLE_TO_LOGIN

デフォルト = 設定値なし

HostName

WebSphere MQ をホスティングしているサーバーの名前です。

デフォルト = 設定値なし

InDoubtEvents

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

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

デフォルト = Reprocess

InputQueue

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

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

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

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

ューを 2 回ずつポーリングしたら、メソッド pollForEvents への呼び出しは完了します。以下に、メッセージ検索の順序を示します。

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

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

InProgressQueue

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

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

PollQuantity

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

デフォルト = 1

Port

WebSphere MQ リスナー用に設定されたポートです。

デフォルト = 設定値なし

ReplyToQueue

コネクターからの要求発行時に応答メッセージが配信されるキューです。子動的メタオブジェクトの属性を使用して応答を無視することもできます。このような属性の詳細については、44 ページの『JMS ヘッダー、WebSphere MQ メッセージ・プロパティー、および動的子メタオブジェクト属性』を参照してください。

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

UnsubscribedQueue

アンサブスクライブされたメッセージが送信されるキューです。

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

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

UseDefaults

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

複数コネクター・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。それには、以下の作業を行う必要があります。

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

新規ディレクトリーの作成

コネクター・インスタンスごとにコネクター・ディレクトリーを作成する必要があります。このコネクター・ディレクトリーには、ご使用のプラットフォームに応じた名前を付けます。

Windows システム

新規ディレクトリーは、*ProductDir*¥connectors¥*connectorInstance* と命名する必要があります。ここで、*connectorInstance* はコネクター・インスタンスを固有に識別する名前です。

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は、コネクター・インスタンス用のメタオブジェクトを作成する必要があります。メタオブジェクトをファイルとして保管する場合は、このファイルを格納するために、*ProductDir*¥repository¥*connectorInstance* ディレクトリーを作成します。

InterChange Server Express サーバー名を `start_WebSphereMQ.bat` ファイルのパラメーターとして指定できます。以下に例を示します。

```
start_WebSphereMQ.bat connName WebSphereICSName
```

OS/400 システム

新規ディレクトリーは、/QIBM/UserData/WBIServer43/*WebSphereICSName*/connectors/*connectorInstance* と

命名する必要があります。ここで、*WebSphereICSName* は InterChange Server Express インスタンスの名前であり、*connectorInstance* はコネクター・インスタンスを固有に識別する名前です。

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は、コネクター・インスタンス用のメタオブジェクトを作成する必要があります。メタオブジェクトをファイルとして保管する場合は、
/QIBM/UserData/WBIServer43/*WebSphereICSName*/repository/*connectorInstance* を作成します。

InterChange Server Express サーバー名を start_WebSphereMQ.sh ファイルのパラメーターとして指定できます。以下に例を示します。

```
start_WebSphereMQ.sh connName WebSphereICSName [-cConfigFile]
```

Linux システム

新規ディレクトリーは、*ProductDir*/connectors/*connectorInstance* と命名する必要があります。ここで、*connectorInstance* はコネクター・インスタンスを固有に識別する名前です。

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は、コネクター・インスタンス用のメタオブジェクトを作成する必要があります。メタオブジェクトをファイルとして保管する場合は、このファイルを保管するために、*ProductDir*/repository/*connectorInstance* ディレクトリーを作成します。

InterChange Server Express サーバー名を connector_manager コマンドのパラメーターとして指定できます。以下に例を示します。

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

ビジネス・オブジェクト定義の作成

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

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

     ProductDir¥repository¥*initialConnectorInstance*

     追加作成したファイルは、*ProductDir*¥repository の適切な *connectorInstance* サブディレクトリーに保管する必要があります。
   - **OS/400 システム:**

     QIBM/UserData/WBIServer43/*WebSphereICSName*/repository
     /*initialConnectorInstance*

     追加作成したファイルは、
     QIBM/UserData/WBIServer43/*WebSphereICSName*/repository/ の適切な *connectorInstance* サブディレクトリーに保管する必要があります。

- **Linux システム:**

  ProductDir/repository/*initialConnectorInstance*

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

コネクター定義の作成

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

始動スクリプトの作成

始動スクリプトを作成するには、次の作業を実行します。

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

   dirname

2. (*Linux* システムのみ) 始動スクリプト CONJAR を CONJAR=${CONDIR}/CW${CONNAME}.jar から以下のように変更します。

   ```
   CONJAR=${CONDIR}/CWWebSphereMQ.jar
   ```

3. この始動スクリプトを 29 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに置く。
4. (*Windows* システムのみ) 始動スクリプトのショートカットを作成します。
5. (*Windows* システムのみ) 初期コネクターのショートカット・テキストをコピーして、新規コネクター・インスタンス名に一致するように初期コネクターの名前を (コマンド行で) 変更します。
6. (*OS/400* システムのみ) 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(QWBIWEBMQC) FROMLIB(QWBISVR43) OBJTYPE(*JOBD)
   TOLIB(QWBISVR43) NEWOBJ(newConName)
   ```

   ここで、*newConName* は新規 WebSphere MQ コネクターのジョブ記述として使用する 10 文字の名前です。

7. (*OS/400* システムのみ) 新規コネクターを Console に追加します。詳しくは、Console に付属のオンライン・ヘルプを参照してください。

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

キューの Uniform Resource Identifiers (URI)

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

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

例えば、次の URI を指定した場合、キュー・マネージャー crossworlds.queue.manager に存在するキュー IN に接続し、すべてのメッセージが優先順位 5 の WebSphere MQ メッセージとして送信されます。

```
queue://crossworlds.queue.manager/MQCONN.IN?targetClient=1&priority=5
```

以下の表に、キュー URI のプロパティー名を示します。

プロパティー名	説明	値
expiry	メッセージの存続時間 (ミリ秒単位)	0 = 無制限。正の整数 = タイムアウト (ミリ秒単位)。
priority	メッセージの優先順位	0 から 9 で、1 が最高の優先順位。値 -1 は、このプロパティーがキューの構成によって決定されることを意味します。値 -2 は、コネクター自身のデフォルト値を使用できるよう指定します。
persistence	メッセージをディスクに「ハード化」するかどうか	1 = 非永続 2 = 永続値 -1 は、このプロパティーがキューの構成によって決定されることを意味します。値 -2 は、コネクター自身のデフォルト値を使用できるよう指定します。
CCSID	アウトバウンド・メッセージをエンコードする文字セット	整数: WebSphere MQ の資料にリストされている有効な値。この値は、CCSID コネクター固有の構成プロパティーの値と一致する必要があります。 25 ページの『CCSID』を参照してください。
targetClient	受信側アプリケーションが JMS 準拠であるかどうか	0 = JMS (MQRFH2 ヘッダー) 1 = MQ (MQMD ヘッダーのみ)
encoding	数値フィールドの表示方法	基本的な WebSphere MQ 資料に記載されている整数値。

注: アダプターは、MQMessage 内のデータの文字セット (CCSID) またはエンコード属性を制御できません。データ変換はデータがメッセージ・バッファーから検索されるかメッセージ・バッファーにデリバリーされるときに行われるため、コネクターは JMS の IBM WebSphere MQ インプリメンテーションに依

存してデータ変換を行います (IBM WebSphere MQ Java クライアント・ライブラリーの資料を参照してください)。したがって、これらの変換は、ネイティブ WebSphere MQ API がオプション MQGMO_CONVERT を使用して実行する変換と双方向で等しくなければなりません。コネクターは、変換プロセスにおける差異または失敗を制御できません。コネクターは、特別な変更を必要とせずに、WebSphere MQ によってサポートされるすべての CCSID またはエンコードのメッセージ・データを検索できます。特定の CCSID またはエンコードのメッセージを送信するには、出力キューが完全修飾 URI であり、CCSID および encoding の値を指定している必要があります。コネクターはこの情報を WebSphere MQ に渡し、WebSphere MQ は MQMessage のデリバリーのためにデータをエンコードするときに (JMS API を介して) この情報を使用します。多くの場合、CCSID およびエンコードのサポートの欠如は、IBM の Web サイトから最新バージョンの IBM WebSphere MQ Java クライアント・ライブラリーをダウンロードすることによって解決できます。それでも CCSID およびエンコードに関する問題が解消されない場合は、WebSphere Business Integration Server Express システムのテクニカル・サポートに連絡し、代替の Java 仮想マシンを使用してコネクターを実行することを検討してください。

メタオブジェクト属性構成

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

- 静的なコネクター・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。

静的メタオブジェクト

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

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

以下の表で、メタオブジェクトのプロパティーを説明します。

プロパティー名	説明
CorrelationID	このプロパティーは要求処理中のアダプター動作にのみ影響を与えます。このプロパティーの処理方法は、動的メタオブジェクト内の CorrelationID プロパティーと同じです。詳しくは、 48 ページの『非同期要求処理』 を参照してください。
CollaborationName	CollaborationName は、ビジネス・オブジェクトと動詞の組み合わせに対する属性のアプリケーション固有のテキスト内で指定される必要があります。例えば、ユーザーが動詞 Create 付きのビジネス・オブジェクト Customer の同期要求を処理しようとしている場合、静的メタデータ・オブジェクトは Customer_Create という名前の属性を含んでいる必要があります。 Customer_Create 属性は、名前と値のペアを含むアプリケーション固有のテキストを含んでいる必要があります。例えば、CollaborationName=MyCustomerProcessingCollab です。構文の詳細については、 36 ページの『アプリケーション固有の情報』のセクションを参照してください。 この条件が満たされていない場合は、コネクターが Customer ビジネス・オブジェクトに関する要求を同期的に処理しようとするとランタイム・エラーが発生します。 **注:** このプロパティーは、同期要求にのみ利用可能です。
DataEncoding	DataEncoding は、メッセージの読み取りおよび書き込みに使用されるエンコードです。このプロパティーが静的メタオブジェクトで指定されていない場合、コネクターは特定のエンコードを使用せずにメッセージを読み取ろうとします。動的子メタオブジェクトで定義された DataEncoding は、静的メタオブジェクトで定義された値をオーバーライドします。デフォルト値は Text です。この属性の値のフォーマットは、messageType[:*enc*] です。例えば、Text:ISO8859_1、Text:UnicodeLittle、Text、または Binary のようになります。このプロパティーは内部的に InputFormat プロパティーに関連します。InputFormat ごとに 1 つの DataEncoding のみを指定します。
DataHandlerConfigMO	構成情報を提供するために、データ・ハンドラーに渡されるメタオブジェクト。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。動的な子メタオブジェクトに指定された場合、このプロパティーはコネクター・プロパティーと静的なメタオブジェクトのプロパティーをオーバーライドします。データ形式が実際のビジネス・データに依存する可能性がある場合は、要求処理には動的な子メタオブジェクトを使用します。指定されるビジネス・オブジェクトは、コネクターによってサポートされている必要があります。

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

プロパティー名	説明
InputQueue	コネクターが、新しいメッセージを検出するためにポーリングする入力キュー。 **注:** コネクター固有のプロパティーにある InputQueue プロパティーは、アダプターがポーリングするキューを定義します。これは、アダプターがポーリングするキューを決定するのに使用する唯一のプロパティーです。静的 MO では、InputQueue プロパティーおよび InputFormat プロパティーは、アダプターが指定されたメッセージを特定のビジネス・オブジェクトにマップする条件として使用できます。データ・ハンドラーの InputQueues へのマッピングについては、37 ページの『データ・ハンドラーから InputQueues へのマッピング』を参照してください。
OutputQueue	OutputQueue は、特定のビジネス・オブジェクトから派生したメッセージが配信される出力キューです。動的な子メタオブジェクトに定義された OutputQueue は、静的なメタオブジェクトに定義された値をオーバーライドします。
ResponseTimeout	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未定義のままか、またはゼロよりも小さい値に設定されている場合、コネクターは応答を待機せず、SUCCESS を即時に戻します。動的な子メタオブジェクトに定義された ResponseTimeout は、静的なメタオブジェクトに定義された値をオーバーライドします。
TimeoutFatal	このプロパティーが定義されていて値が True の場合、ResponseTimeout で指定された時間内に応答がないと、コネクターは APP_RESPONSE_TIMEOUT を戻します。応答メッセージを待機しているその他のすべてのスレッドは、InterChange Server Express に APP_RESPONSE_TIMEOUT を即時に戻します。これにより、InterChange Server Express はコネクターを終了します。動的な子メタオブジェクトに定義された TimeoutFatal は、静的なメタオブジェクトに定義された値をオーバーライドします。

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

以下に、サンプル・メタオブジェクトを示します。

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

アプリケーション固有の情報

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

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

データ・ハンドラーから InputQueues へのマッピング

静的メタオブジェクトのアプリケーション固有情報で InputQueue プロパティーを使用することにより、データ・ハンドラーと入力キューを関連付けることができます。この機能は、異なる書式や変換要件を持つ複数の取引先と取り引きする場合に役立ちます。それには、以下の作業を行う必要があります。

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

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

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

入力フォーマットの多重定義

コネクターは通常、メッセージ検索時に入力フォーマットを特定のビジネス・オブジェクトと動詞の組み合わせと付き合わせます。次に、コネクターはそのビジネス・オブジェクト名とメッセージの内容をデータ・ハンドラーに渡します。これにより、データ・ハンドラーは、メッセージの内容がユーザーの要求するビジネス・オブジェクトと対応しているかどうかを確認できます。

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

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

サンプル・メタオブジェクト

以下に示す静的メタオブジェクトは、動詞 Create、Update、Delete、および Retrieve を使用する Customer ビジネス・オブジェクトを変換するコネクターを構成します。属性 Default はメタオブジェクトで定義されます。コネクターは以下の属性を持つ変換プロパティーを使用します。

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

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

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

動詞 Update および Delete 付き Customer オブジェクト: 入力フォーマット MODIFY は多重定義されています。この入力フォーマットは、動詞 Update 付きのビジネス・オブジェクト Customer と動詞 Delete 付きのビジネス・オブジェクト Customer の両方に対して定義されています。検索されたこのフォーマットのメッセージを正常に処理するためには、メッセージの内容にビジネス・オブジェクトと動詞が含まれ、データ・ハンドラーがそれらを識別できるようになっている必要があります (37 ページの『入力フォーマットの多重定義』を参照してください)。Request 処理操作では、出力フォーマットが定義されていないため、コネクターは入力フォーマット MODIFY を使用していずれかの動詞のメッセージを送信します。

動詞 Retrieve 付き Customer オブジェクト: 属性 Customer_Retrieve は、動詞 Retrieve 付きの Customer タイプのビジネス・オブジェクトを、フォーマット Retrieve のメッセージとして送信する必要があることを指定します。デフォルトの応答時間は、コネクターがタイムアウトまでに最大 10000 ミリ秒待機するようにオーバーライドされているので注意してください (応答が受信されない場合も終了します)。

```
[ReposCopy]
Version = 3.1.0
Repositories = 1cHyILNuPTc=
[End]
[BusinessObjectDefinition]
Name = Sample_MO
Version = 1.0.0

[Attribute]
Name = Default
Type = String
Cardinality = 1
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
OutputQueue=CustomerQueue1;ResponseTimeout=5000;TimeoutFatal=true
```

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

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

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

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

動的な子メタオブジェクト

静的メタオブジェクトに必要なメタデータを指定することが困難または実行不可能な場合、オプションで、コネクターが実行時にビジネス・オブジェクト・インスタンスごとに指定されたメタデータを受け入れることができます。

コネクターは、コネクターに渡されるトップレベルのビジネス・オブジェクトに子として追加された動的メタオブジェクトから、変換プロパティーを認識して読み取ります。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。

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

前のセクションの表と次の表に、ビジネス・オブジェクト (Customer_Create) の静的子メタオブジェクトと動的子メタオブジェクトのサンプルをそれぞれ示します。アプリケーション固有の情報は名前と値のペアで構成され、それぞれのペアはセミコロンで区切られています。

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

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

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

ポーリング中の動的な子メタオブジェクトの取り込み

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

次の表に、動的子メタオブジェクトのポーリングのための構成の例を示します。

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

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

シナリオ例:

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

動的子メタオブジェクトのサンプル

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

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

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

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

[Attribute]
Name = FirstName
Type = String
```

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

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

JMS ヘッダー、WebSphere MQ メッセージ・プロパティー、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。このような属性を追加すると、JMS プロパティーを変更し、(アダプター・プロパティーで指定されたデフォルト ReplyToQueue を使用せずに) 要求ごとに ReplyToQueue を制御したり、メッセージの CorrelationID を再ターゲットしたりすることができます。このセクションでは、これらの属性、および同期モードと非同期モードの両方におけるイベント通知と要求処理に対する影響について説明します。

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

表 4. 動的メタオブジェクト・ヘッダー属性

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

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

以下のセクションでは、これらの属性の解釈および使用について説明します。

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

JMS プロパティー: 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。

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 5. JMS プロパティー属性のアプリケーション固有情報

名前	指定可能な値	コメント
名前	任意の有効な JMS プロパティー名	これは JMS プロパティーの名前です。ベンダーによっては、拡張機能を提供するために特定のプロパティ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、Int、Boolean、Float、Double、Long、Short	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。

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

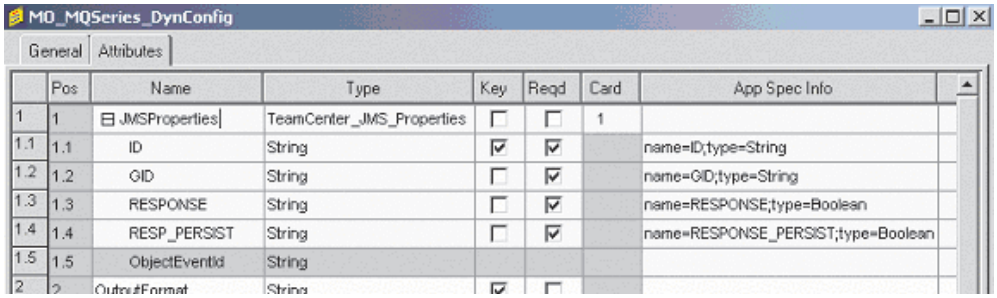

図 3. 動的メタオブジェクトの JMS プロパティー属性

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

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

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

同期イベント通知: 同期イベント処理の場合は、アダプターはイベントを通知し、InterChange Server Express からの応答を待った後、アプリケーションに応答メッセージを送信します。ビジネス・データに対する変更は、戻される応答メッセージに反映されます。イベントを通知する前に、アダプターは、非同期イベント通知の場合と同様に動的メタオブジェクトを設定します。動的メタオブジェクトに設定される値は、以下のように応答発行ヘッダーに反映されます (動的メタオブジェクトの他の読み取り専用属性は無視されます)。

- **CorrelationID** 動的メタオブジェクトが属性 CorrelationId を含む場合は、発信元アプリケーションが必要とする値に設定する必要があります。アプリケーションは、CorrelationID を使用してコネクターから戻されたメッセージと元の要求を突き合わせます。CorrelationID が予期しない値または無効値の場合は、問題が発生します。これは、この属性を使用する前にアプリケーションが相関する要求および応答メッセージを処理する方法を判別するのに役立ちます。同期要求で CorrelationID を設定するには 4 つの方法があります。
  1. 値を変更しない。応答メッセージの CorrelationID は、要求メッセージの CorrelationID と同じになります。これは、WebSphere MQ オプション MQRO_PASS_CORREL_ID と同等です。
  2. 値を CxIgnore に変更する。コネクターは、デフォルトで要求のメッセージ ID を応答の CorrelationID にコピーします。これは、WebSphere MQ オプション MQRO_COPY_MSG_ID_TO_CORREL_ID と同等です。
  3. 値を CxBlank に変更する。コネクターは応答メッセージの CorrelationID を設定しません。
  4. 値をカスタム値に変更する。これを行うには、応答を処理するアプリケーションでカスタム値を認識する必要があります。

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

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

1. 属性 CorrelationID が動的メタオブジェクトに存在する場合は、コネクターは、アウトバウンド要求メッセージの CorrelationID をこの値に設定します。
2. 属性 ReplyToQueue が動的メタオブジェクトで指定されている場合は、コネクターは、応答メッセージでこのキューを渡し、このキューで応答を待ちます。これにより、コネクター構成プロパティーで指定されている ReplyToQueue 値をオーバーライドできます。さらに負の ResponseTimeout (コネクターが応答を待たないことを示す) を指定した場合は、コネクターは実際には応答を待ちませんが、応答メッセージで ReplyToQueue が設定されます。
3. 属性 DeliveryMode を 2 に設定すると、メッセージは永続的に送信されます。DeliveryMode を 1 に設定すると、メッセージは永続的に送信されません。その他の値を設定すると、コネクターに障害が発生します。MO に DeliveryMode を指定しないと、JMS プロバイダーが永続設定を確立します。
4. 属性 Priority を指定すると、コネクターが発信要求に値を設定します。Priority 属性には 0 から 9 までの値を設定できます。その他の値を指定すると、コネクターが終了します。
5. 動的メタオブジェクトで属性 JMSProperties を指定した場合は、コネクターによって送信されるアウトバウンド・メッセージに、子動的メタオブジェクトで指定された対応する JMS プロパティーが設定されます。

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

同期要求処理: コネクターは、動的メタオブジェクト (存在する場合) を使用して要求メッセージを設定してから発行します。動的メタオブジェクトがヘッダー属性を含む場合は、コネクターは、応答メッセージで検出された対応する新しい値をそのヘッダー属性に設定します。コネクターは、応答メッセージを受信した後、(メタオブジェクトにトランスポート関連のデータを設定するほかに) 以下のステップを実行します。

1. 属性 CorrelationID が動的メタオブジェクトに存在する場合は、アダプターは、応答メッセージで指定された JMSCorrelationID でこの属性を更新します。
2. 属性 ReplyToQueue が動的メタオブジェクトで定義されている場合は、アダプターは、応答メッセージで指定された JMSReplyTo の名前でこの属性を更新します。
3. 属性 DeliveryMode が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSDeliveryMode ヘッダー・フィールドの値でこの属性を更新します。
4. 属性 Priority が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSPriority ヘッダー・フィールドの値でこの属性を更新します。
5. 属性 Destination が動的メタオブジェクトで定義されている場合は、アダプターは、応答メッセージで指定された JMSDestination の名前でこの属性を更新します。

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

注: 動的メタオブジェクトを使用して要求メッセージで設定された CorrelationID を変更しても、アダプターが応答メッセージを識別する方法には影響しません。アダプターは、デフォルトですべての応答メッセージの CorrelationID がアダプターによって送信された要求のメッセージ ID に等しいことを要求します。

エラー処理: JMS プロパティーをメッセージから読み取れない場合、またはメッセージに書き込めない場合は、コネクターはエラーをログに記録し、要求またはイベントは失敗します。ユーザー指定の ReplyToQueue が存在しないかアクセスできない場合は、コネクターはエラーをログに記録し、要求は失敗します。CorrelationID が無効であるか設定できない場合は、コネクターはエラーをログに記録し、要求は失敗します。いずれの場合も、ログに記録されたメッセージはコネクターのメッセージ・ファイルからのものです。

始動

コネクターの始動

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

ProductDir¥connectors¥*connName*

このディレクトリー名の *connName* はコネクターを表しています。表 6 が示すとおり、始動スクリプトの名前はオペレーティング・システム・プラットフォームにより異なります。

表 6. 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	start_*ConnName*.sh **注:** start_connName.sh 始動スクリプトを手動で実行する必要はありません。代わりに、以下のコマンドを使用して必要な環境変数を設定し、始動スクリプトを自動的に起動します。 connector_manager -start ***connName*** [-cConfigFile]

始動スクリプトの起動 (Windows システムの場合)

以下のいずれかの方法で、コネクター始動スクリプトを起動できます。

- **Windows の「スタート」メニューから**

  「プログラム」->「IBM WebSphere Business Integration Express」->「Adapters」->「Connectors」-> 「***your_connector_name***」をクリックします。

  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。コネクターのデスクトップ・ショートカットを作成することもできます。

- **コマンド行から**

  以下のコマンドを入力します。

  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 (デフォルトでは WebSphereICS) です。

- **Windows のサービスとして**

  コネクターを Windows のサービスとして始動するように構成できます。この場合、自動サービスでは Windows システムがブートするとき、手動サービスでは

「Windows サービス」ダイアログ・ボックスからサービスを開始するときに、コネクターが開始されます。デフォルトでは、サービス名は CWConnector WBIWebServicesAdapter です。

- **System Monitor から**

  System Monitor を介して、コネクターをロード、活動化、非活動化、休止、シャットダウン、または削除できます。

始動スクリプトの起動 (OS/400 システムの場合)

以下のいずれかの方法で、コネクター始動スクリプトを起動できます。

- **Windows システムから**

  以下の手順を実行します。

  1. WebSphere Business Integration Console がインストールされているマシンから、**「スタート」->「プログラム」->「IBM WebSphere Business Integration Console」->「Console」**をクリックします。
  2. コネクターがインストールされているマシンの OS/400 システム名または IP アドレスを指定します。
  3. *JOBCTL 権限を持つユーザーのユーザー・プロファイルおよびパスワードを指定します。
  4. リストから適切なコネクターを選択し、**「アダプターを始動」**をクリックします。

- **OS/400 コマンド行から**

  以下のように、バッチ・モードまたは対話モードを使用します。

  – *バッチ・モード*

    CL コマンド QSH を実行し、QSHELL 環境を起動します。QSHELL 内から、以下のスクリプトを実行します。

    ```
    /QIBM/ProdData/WBIServer43/bin/submit_adapter.sh connName
    WebSphereICSName pathToConNameStartScript jobDescriptionName
    ```

    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、*pathToConNameStartScript* 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されるジョブ記述名です。

  – *対話モード*

    CL コマンド QSH を実行し、QSHELL 環境を起動します。QSHELL 内から、以下のスクリプトを実行します。

    ```
    /QIBM/UserData/WBIServer43/WebSphereICSName/connectors ¥
    /connName/start_connName.sh connName WebSphereICSName [-cConfigFile]
    ```

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

- **System Monitor から**

  System Monitor を介して、コネクターをロード、活動化、非活動化、休止、シャットダウン、または削除できます。

注: OS/400 システム上で自動的に始動された TCP/IP サーバーを使用してコネクターを始動するには、以下のスクリプトを使用します。

```
/QIBM/ProdData/WBIServer43/bin/add_autostart_adapter.sh connName
WebSphereICSName pathToConNameStartScript jobDescriptionName
```

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、*pathToConNameStartScript* 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されるジョブ記述名です。

始動スクリプトの起動 (Linux システムの場合)

Linux システム上で始動スクリプトを起動するには、以下のコマンドを入力します。

```
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ために使用する方法は、ご使用のプラットフォームによって異なります。以下のいずれかのセクションを参照してください。

コネクターの停止 (Windows システムの場合)

以下のいずれかの方法でコネクターを停止できます。

- **コネクター・プロセスを実行しているウィンドウから**

  「コネクター」ウィンドウで q を押します。

- **Windows の「サービス」ダイアログ・ボックスから**

  コネクターが Windows のサービスとして構成されている場合は、Windows の「サービス」ダイアログ・ボックスでサービスを停止できます。

- **System Monitor から**

  System Monitor を介して、コネクターをロード、活動化、非活動化、休止、シャットダウン、または削除できます。

コネクターの停止 (OS/400 システムの場合)

以下のいずれかの方法でコネクターを停止できます。

- **Console またはコマンド行から**

  Console を使用してコネクターを始動した場合、あるいは、QSHELL 環境で 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、以下を実行してコネクターを停止します。

  1. CL コマンド WRKACTJOB SBS(QWBISVR43) を使用して、WebSphere Business Integration Server Express のジョブを表示します。

2. リストをスクロールして、ご使用のコネクターのジョブ記述 (例えば、WebSphere MQ コネクターのジョブ記述は QWBIWEBMQC) に一致するジョブ名を持つジョブを探し出します。
3. このジョブに対してオプション 4 を選択します。
4. F4 を押して、ENDJOB コマンドのプロンプトを表示します。
5. 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。

QSHELL 環境から start_*connName*.sh スクリプトを使用してコネクターを始動した場合は、F3 を押してコネクターを停止します。

- **System Monitor から**

  System Monitor を介して、コネクターをロード、活動化、非活動化、休止、シャットダウン、または削除できます。

コネクターの停止 (Linux システムの場合)

コネクターは、Linux システムのバックグラウンドで実行されます。コネクター用の個別のウィンドウはありません。コネクターを停止するには、以下のコマンドを実行します。

```
connector_manager -stop connName WebSphereICSName
```

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。

第 3 章 ビジネス・オブジェクトの作成および変更

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

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

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

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

アダプターのビジネス・オブジェクトの構造

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

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

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

ビジネス・オブジェクト・プロパティーのサンプル

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

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

[Attribute]
Name = ContactId
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = true
DefaultValue = 1001
IsRequiredServerBound = false
[End]
[Attribute]
Name = FirstName
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = Jim
IsRequiredServerBound = false
[End]
[Attribute]
Name = LastName
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = Smith
IsRequiredServerBound = false
[End]
[Attribute]
Name = OfficePhoneArea
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = 650
IsRequiredServerBound = false
[End]
[Attribute]
Name = OfficePhone
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
```

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

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

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

エラー処理

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

```
Message number
Message text
```

コネクターは、以降のセクションで説明する方法で特定のエラーを処理します。

アプリケーション・タイムアウト

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

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

アンサブスクライブされたビジネス・オブジェクト

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

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

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

コネクターがアクティブでない

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

データ・ハンドラーによる変換

データ・ハンドラーがメッセージからビジネス・オブジェクトへの変換に失敗した場合、または、(JMS プロバイダーではなく) ビジネス・オブジェクトに固有の処理エラーが発生した場合、メッセージは ErrorQueue で指定されたキューにデリバリーされます。ErrorQueue が定義されていない場合、エラーが原因で処理できなかったメッセージは破棄されます。

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

トレース

トレースはオプションのデバッグ機能で、オンにするとコネクターの動作を詳細にトレースできます。デフォルトでは、トレース・メッセージは STDOUT に書き込まれます。トレース・メッセージの構成の詳細については、19 ページの『第 2 章 アダプターのインストールおよび構成』のコネクター構成プロパティーの説明を参照してください。トレースの使用可能化や設定方法などの詳細については、「システム*管理ガイド*」を参照してください。

コネクターのトレース・メッセージに推奨される内容を以下に示します。

レベル 0　コネクターのバージョンを確認するトレース・メッセージに使用します。

レベル 1　処理される各ビジネス・オブジェクトについての重要な情報を提供するトレース・メッセージや、ポーリング・スレッドが入力キュー内で新しいメッセージを検出するたびに記録されるトレース・メッセージに使用します。

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

レベル 3　メッセージからビジネス・オブジェクトへの変換およびビジネス・オブジェクトからメッセージへの変換に関する情報を提供するトレース・メッセージや、出力キューへのメッセージのデリバリーに関する情報を提供するトレース・メッセージに使用します。

レベル 4　コネクターが動作を開始または終了した時間を記録するトレース・メッセージに使用します。

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

第 4 章 トラブルシューティング

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

注: MQJMS2005 を除き、以下にリストしたすべてのエラーは、Windows 版のアダプターの実行中にのみ発生します。

始動時の問題

問題	考えられる処置/説明
初期化中にコネクターが予期しないエラーでシャットダウンし、次のメッセージが報告されました: `Exception in thread "main" java.lang.NoClassDefFoundError: javax/jms/JMSException...`	コネクターは、IBM WebSphere MQ Java クライアント・ライブラリーからファイル `jms.jar` を見つけることができません。`CWConnEnv.bat` ファイルまたは `CWConnEnv.sh` ファイルの変数 `MQ_LIB` が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認します。
初期化中にコネクターが予期しないエラーでシャットダウンし、次のメッセージが報告されました: `Exception in thread "main" java.lang.NoClassDefFoundError: com/ibm/mq/jms/MQConnectionFactory...`	コネクターは、IBM WebSphere MQ Java クライアント・ライブラリーから `com.ibm.mqjms.jar` ファイルを見つけることができません。`CWConnEnv.bat` ファイルまたは `CWConnEnv.sh` ファイルの変数 `MQ_LIB` が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認します。
初期化中にコネクターが予期しないエラーでシャットダウンし、次のメッセージが報告されました: `Exception in thread "main" java.lang.NoClassDefFoundError: javax/naming/Referenceable...`	コネクターは、IBM WebSphere MQ Java クライアント・ライブラリーからファイル `jndi.jar` を見つけることができません。`CWConnEnv.bat` ファイルまたは `CWConnEnv.sh` ファイルの変数 `MQ_LIB` が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認します。
初期化中にコネクターが予期しないエラーでシャットダウンし、次の例外が報告されました: `java.lang.UnsatisfiedLinkError: no mqjbnd01 in shared library path`	コネクターは、IBM WebSphere MQ Java クライアント・ライブラリーから必要なランタイム・ライブラリー (`mqjbnd01.dll` [NT] または `libmqjbnd01.so` [Solaris]) を見つけることができません。パスに IBM WebSphere MQ Java クライアント・ライブラリーのフォルダーが含まれていることを確認します。
コネクターから次の例外が報告されました: `MQJMS2005: failed to create MQQueueManager for ':'`	次のプロパティーの値を明示的に設定します: `HostName`、`Channel`、および `Port`。

イベント処理

問題	考えられる処置/説明
コネクターは、MQRFH2 ヘッダーを持つすべてのメッセージをデリバリーします。	MQMD WebSphere MQ ヘッダーを持つメッセージのみをデリバリーするには、出力キューの URI の名前に `?targetClient=1` を付加します。例えば、メッセージをキュー `queue://my.queue.manager/OUT` に出力する場合は、URI を `queue://my.queue.manager/OUT?targetClient=1` に変更します。詳細については、19 ページの『第 2 章 アダプターのインストールおよび構成』を参照してください。
コネクターは、コネクター・メタオブジェクト内でのメッセージ・フォーマットの定義にかかわらず、デリバリー時にすべてのメッセージ・フォーマットの 8 文字を超える部分を切り捨てます。	これは WebSphere MQ MQMD メッセージ・ヘッダーの制限であり、コネクターの制限ではありません。

付録 A. コネクターの標準構成プロパティー

この付録では、WebSphere 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 B『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 7 は、標準コネクター構成プロパティーの早見表です。すべてのコネクターがこれらのすべてのプロパティーを使用しているわけではありません。標準プロパティーの依存関係は RepositoryDirectory に基づいているため、プロパティーの設定は統合ブローカーごとに異なります。

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 7. 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	
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント再始動	Repository Directory は <REMOTE>

表 7. 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
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 7. 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 が指定されている
PollStartTime	HH:MM(HH は 0 から 23、MM は 0 から 59)	HH:MM	コンポーネント再始動	
RepositoryDirectory	メタデータ・リポジトリーの場所		エージェント再始動	<REMOTE> に設定する

表 7. 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 の場合のみ適用できます。

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

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

自動およびリモートの再始動機能を使用可能にするには、このプロパティーを `true` に設定する必要があります。MQ により起動される 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 によりトリガーされる OAD がコネクターの再始動を自動的に試行する回数の最大数を指定します。`OADAutoRestartAgent` プロパティーを有効にするには、値を `true` に設定する必要があります。

デフォルト値は `1000` です。

OADRetryTimeInterval

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

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 の場合のみ適用できます。

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。指定できる値の範囲は 1 から 2147483647 です。

デフォルト値は 1 です。

SourceQueue

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

JMS イベント・ストアを使用する JMS 対応コネクタ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、69 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用できます。

同期応答を要求する要求メッセージを、コネクター・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。

デフォルト値は CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用できます。

同期要求に対する応答として送信される応答メッセージを、ブローカーからコネクター・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。

デフォルト値は CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用できます。

コネクタ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**を作成します。
- **コネクター構成ファイル**を作成します。インストールするコネクターごとに 1 つ構成ファイルを作成する必要があります。
- 構成ファイルのプロパティーを設定します。
  場合によっては、コネクター・テンプレートでプロパティーに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、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、81 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

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

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 から実行

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 (86 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、81 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でこのテンプレートを保管します。

新規テンプレートの作成

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

テンプレートは以下のように作成します。

1. **「ファイル」>「新規」>「コネクター固有プロパティー・テンプレート (Connector-Specific Property Template)」**とクリックします。
2. 以下のフィールドを含む**「コネクター固有プロパティー・テンプレート (Connector-Specific Property Template)」**ダイアログ・ボックスが表示されます。
   - **「テンプレート」**および**「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」**および**「変更する既存のテンプレートを選択してください」**

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

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

一般特性の指定

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

- **一般:**
  プロパティー・タイプ
  更新されたメソッド
  説明
- **フラグ**
  標準のフラグ
- **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。また、編集可能な値も設定できます。これを行うには、以下のステップを実行します。

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

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

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

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

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

依存関係の設定

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

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

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

   == (等しい)

   != (等しくない)

   > (より大)

   < (より小)

   >= (より大か等しい)

   <= (より小か等しい)
4. **「条件値」**フィールドで、依存プロパティーをテンプレートに組み込むために必要な値を入力します。
5. **「使用可能なプロパティー」**表示で強調表示された依存プロパティーで、矢印をクリックし、**「依存プロパティー」**表示に移動させます。
6. **「完了 (Finish)」**をクリックします。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、そのテンプレートを使用して構成ファイルを作成できます。

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

     コネクターの名前を入力します。名前では大文字と小文字が区別されます。入力する名前は、システムにインストールされているコネクターのファイル名と一貫性をもつ一意の名前である必要があります。

     重要: Connector Configurator 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 が実行中である必要があります。
   ファイルとして保管する場合は、**「ファイル・コネクターを保管」**ダイアログ・ボックスが表示されます。`*.cfg` をファイル・タイプとして選択し、「ファイル名」フィールド内に名前が正しいスペル (大文字と小文字の区別を含む) で表示されていることを確認してから、ファイルを保管するディレクトリーにナビゲートし、**「保管」**をクリックします。Connector Configurator 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」**とクリックします。

構成ファイルの完成

構成ファイルを開くか、プロジェクトからコネクターを開くと、「Connector Configurator 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。

- 87 ページの『標準コネクター・プロパティーの設定』
- 87 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』
- 88 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』
- 90 ページの『関連付けられたマップ』
- 91 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、 92 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プロパティーは、コネクターのアプリケーション固有コンポーネント (アプリケーションと直接対話するコンポーネント) のみに適用されます。各コネクターには、そのコネクターのアプリケーションだけで使用されるアプリケーション固有のプロパティーがあります。これらのプロパティーには、デフォルト値が用意されているものもあれば、そうでないものもあります。また、

一部のデフォルト値は変更することができます。各アダプター・ガイドのインストールおよび構成の章に、アプリケーション固有のプロパティーおよび推奨値が記述されています。

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

- 背景がグレーのフィールドは、標準のプロパティーを表します。値を変更することはできますが、名前の変更およびプロパティーの除去はできません。
- 背景が白のフィールドは、アプリケーション固有のプロパティーを表します。これらのプロパティーは、アプリケーションまたはコネクターの特定のニーズによって異なります。値の変更も、これらのプロパティーの除去も可能です。
- 「値」フィールドは構成可能です。
- 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る場合にはメニューから値を選択します。
3. 標準のプロパティーの値をすべて入力すると、以下のいずれかを実行することができます。
   - 変更内容を破棄し、元の値を保持したままで Connector Configurator 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きに、すべてのカテゴリーで入力した値を一括して保管するかまたは破棄するかを確認するプロンプトが出されます。
   - 修正した値を保管するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「はい」**をクリックします。「ファイル」メニューまたはツールバーから**「保管」>「ファイルに」**をクリックする方法もあります。

アプリケーション固有の構成プロパティーの設定

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

1. グリッドの左上端の部分で右マウス・ボタン・クリックします。ポップアップ・メニュー・バーが表示されます。**「追加」**をクリックしてプロパティーを追加します。子プロパティーを追加するには、親行番号を右マウス・ボタン・クリックして、**「子を追加」**をクリックします。
2. プロパティ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. 87 ページの『標準コネクター・プロパティーの設定』で説明したように、変更内容を保管するかまたは破棄するかを選択します。

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

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

コネクター・プロパティーの暗号化

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

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

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

更新メソッド

付録 A 『コネクターの*標準構成プロパティー*』の 63 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 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」**リストの空のフィールドをクリックします。System Manager プロジェクトに存在するすべてのビジネス・オブジェクト定義を示すドロップダウン・リストが表示されます。
2. 追加するビジネス・オブジェクトをクリックします。
3. ビジネス・オブジェクトの**「エージェント・サポート」**(以下で説明) を設定します。
4. 「Connector Configurator 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・オブジェクトにエージェント・サポートがある場合、システムは、コネクター・エージェントを介してアプリケーションにデータを配布する際にそのビジネス・オブジェクトの使用を試みます。

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

ビジネス・オブジェクトがコネクター・エージェントによってサポートされるよう指定するには、**「エージェント・サポート」**ボックスにチェックマークを付けます。「Connector Configurator 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、関連付けられたマップを明示的にバインドすることが必要になります。

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

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

System Manager でのプロジェクトの使用方法と、配置の詳細については、「*User Guide for IBM WebSphere Business Integration Server Express*」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**「ロケール」**標準構成プロパティーのドロップ・リストに表示されるのは、サポートされる値の一部のみです。ドロップ・リストに、サポートされる他の値を追加するには、製品ディレクトリーの ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。

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

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

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

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。

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

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

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。

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