IBM WebSphere Business Integration
Adapters

Adapter for Telcordia ユーザーズ・ガイド

バージョン 2.6.x

IBM WebSphere Business Integration
Adapters

Adapter for Telcordia ユーザーズ・ガイド

バージョン 2.6.x

お願い

本書および本書で紹介する製品をご使用になる前に、 105 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere Business Integration Adapter for Telcordia (5724-G86) バージョン 2.6.x、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： IBM WebSphere Business Integration Adapters
Adapter for Telcordia User Guide
Version 2.6.x

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.7

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

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

目次

本書について

IBM(R) WebSphere(R) Business Integration Adapter ポートフォリオは、主要な e-business テクノロジー、エンタープライズ・アプリケーション、レガシー、およびメインフレーム・システムに統合コネクティビティーを提供します。製品セットには、ビジネス・プロセスの統合に向けてコンポーネントをカスタマイズ、作成、および管理するためのツールとテンプレートが含まれています。

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

対象読者

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

本書の前提条件

本書の読者は、WebSphere Business Integration システム、ビジネス・オブジェクトとコラボレーションの開発、WebSphere MQ アプリケーション、および Telcorida 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。詳細については、『関連文書』を参照してください。

関連文書

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

以下のサイトから、関連資料をインストールすることができます。

- 一般的なアダプター情報が必要な場合、アダプターを WebSphere Message Broker (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) とともに使用する場合、およびアダプターを WebSphere Application Server とともに使用する場合は、以下のサイトを参照してください。
  - http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- アダプターを InterChange Server とともに使用する場合は、以下のサイトを参照してください。
  - http://www.ibm.com/websphere/integration/wicserver/infocenter
  - http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- Message Broker (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) の詳細については、以下のサイトを参照してください。

– http://www.ibm.com/software/integration/mqfamily/library/manualsa/

- WebSphere Application Server の詳細については、以下を参照してください。
  – http://www.ibm.com/software/webservers/appserv/library.html

上記のサイトには資料のダウンロード、インストール、および表示に関する簡単な説明が記載されています。

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。これらの情報は、WebSphere Business Integration Support Web サイト (http://www.ibm.com/software/integration/websphere/support/) にあります。関心のあるコンポーネント・エリアを選択し、「Technotes」セクションと「Flashes」セクションを参照してください。また、IBM Redbooks (http://www.redbooks.ibm.com/) にもその他の有効な情報があることがあります。

表記上の規則

本書では、以下のような規則を使用しています。

`Courier フォント`	コマンド名、ファイル名、入力情報、システムが画面に出力した情報など、記述されたとおりの値を示します。
太字	初出語を示します。
イタリック、*イタリック*	変数名または相互参照を示します。
青のアウトライン	オンラインで表示したときにのみ見られる青のアウトラインは、相互参照用のハイパーリンクです。アウトラインの内側をクリックすると、参照先オブジェクトにジャンプします。
{ }	構文の記述行の場合、中括弧 {} で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
[ ]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプションのパラメーターです。
...	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、`option[,...]` は、複数のオプションをコンマで区切って指定できることを意味します。
< >	命名規則では、不等号括弧は名前の個々の要素を囲み、各要素を区別します。(例: `<server_name><`*`connector_name`*`>tmp.log`)
ProductDir	IBM WebSphere Business Integration Adapters 製品がインストールされるディレクトリーを表します。
/、¥	本書では、ディレクトリー・パスに円記号 (¥) を使用します。UNIX システムの場合には、円記号をスラッシュ (/) に置き換えてください。すべての製品パス名は、システム上の、製品のインストール先ディレクトリーを基準とした相対パス名です。
%text% および *$text*	% 記号で囲まれたテキストは、Windows(R) の `text` システム変数またはユーザー変数の値を示します。UNIX 環境での同等の表記は *$text* です。これは、UNIX 環境変数 *text* の値を示します。

本リリースの新機能

リリース 2.6.x の新機能

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

同期要求に対する応答メッセージを処理する際、コネクターはフィードバック・コード MQFB_NONE (設定されていない場合のデフォルトのフィードバック・コード) を VALCHANGE として解釈します。詳細については、 14 ページの『同期の引き渡し』を参照してください。

このリリースのアダプターは、Telcordia Service Delivery 11.0 をサポートします。

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

リリース 2.5.x の新機能

バージョン 2.5.x から、Adapter for Telcordia は Microsoft Windows NT 上ではサポートされなくなりました。

アダプターのインストール情報は、本書から移動しました。この情報の新たな入手先については、 23 ページの『第 2 章 コネクターのインストールと構成』を参照してください。

リリース 2.4 の新機能

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

アダプターで、Telcordia Service Delivery 10.0 がサポートされました。

コネクターは、以下のプラットフォーム上で実行されます。

- Solaris 7、8
- AIX 5.x

リリース 2.3.x の新機能

「CrossWorlds」という名前を使用してシステム全体を示したりコンポーネントまたはツールの名前を変更したりすることはなくなりましたが、その他の点では以前とほとんど同じです。例えば「CrossWorlds System Manager」は「System Manager」となり、「CrossWorlds InterChange Server」は「WebSphere InterChange Server」となっています。

データ・ハンドラーを入力キューと関連付けることができるようになりました。詳細については、 39 ページの『データ・ハンドラーの入力キューへのマッピングの概要』を参照してください。

保証付きイベント・デリバリー機能が拡張されました。

リリース 2.2.x の新機能

InProgress キューは不要になりました。使用不可に設定できます。詳細については、 30 ページの『InProgressQueue』を参照してください。

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

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

データ・ハンドラーが明示的にビジネス・オブジェクトに対して動詞を割り当てていない場合、コネクターがデフォルトの動詞を適用できるようになりました。詳細については、 28 ページの『DefaultVerb』を参照してください。

ReplyToQueue は、ReplyToQueue コネクター・プロパティーではなく動的子メタオブジェクトを介して指定できるようになりました。詳細については、 42 ページの『JMS ヘッダーおよび動的子メタオブジェクト属性』を参照してください。

メッセージ選択子を使用して、識別やフィルター操作を行えます。あるいは、アダプターが特定の要求に対して応答メッセージを識別する方法を制御できます。この JMS 機能は同期要求処理にのみ適用されます。詳細については、 14 ページの『同期の引き渡し』を参照してください。

同一スキーマに組み込まれているファイルのいくつかに同様の名前のエレメント定義が含まれる場合に、そのスキーマからビジネス・オブジェクトを生成する方法については、 52 ページの『スキーマ文書に基づく Telcordia ビジネス・オブジェクトの生成』を参照してください。

第 1 章 概要

Connector for Telcordia™[1] は、IBM WebSphere Business Integration Adapter for Telcordia のランタイム・コンポーネントです。このコネクターを使用すると、WebSphere 統合ブローカーと、Telcordia 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るようになります。本章では、コネクター・コンポーネントと、これに関係するビジネス・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ます。コネクター・フレームワークは統合ブローカーとアプリケーション固有のコンポーネントの間の仲介役として機能し、そのコードはどのコネクターにも共通です。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントとの間で以下のようなサービスを提供します。

- ビジネス・オブジェクトの送受信
- 始動メッセージおよび管理メッセージの交換の管理

本書では、アプリケーション固有のコンポーネントとコネクター・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。

アダプター環境

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

1. Telcordia Technologies, Inc.

ブローカーの互換性

アダプターが使用するアダプター・フレームワークは、アダプターと通信する統合ブローカーのバージョンとの互換性を備えている必要があります。Adapter for EJB バージョン 2.6.x は、以下のバージョンのアダプター・フレームワークおよび統合ブローカーでサポートされています。

- アダプター・フレームワーク:
  - WebSphere Business Integration Adapter Framework、バージョン 2.1.0、 2.2.0、2.3.0、 2.3.1、 2.4.0
- 統合ブローカー:
  - WebSphere InterChange Server、バージョン 4.1.1、4.2.0、4.2.1、4.2.2
  - WebSphere MQ Integrator、バージョン 2.1
  - WebSphere MQ Integrator Broker、バージョン 2.1.0
  - WebSphere Business Integration Message Broker、バージョン 5.0
  - WebSphere Application Server Enterprise、バージョン 5.0.2 (WebSphere Studio Application Developer Integration Edition、バージョン 5.0.1 と併用)

例外については、「*リリース情報*」を参照してください。

注: 統合ブローカーのインストール手順およびその前提条件については、次の資料を参照してください。

WebSphere InterChange Server (ICS) の場合は、「*システム・インストール・ガイド (UNIX 版)*」また は「*システム・インストール・ガイド (Windows 版)*」を参照してください。

Message Brokers (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) の場合は、「*WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド*」およびそれぞれの Message Brokers のインストールに関する資料を参照してください。一部は次の Web サイトで入手可能です。

http://www.ibm.com/software/integration/mqfamily/library/manualsa/

WebSphere Application Server については、「アダプター実装ガイド (WebSphere Application Server)」および次の場所にある資料を参照してください。

http://www.ibm.com/software/webservers/appserv/library.html

アダプターのプラットフォーム

このアダプターは、次のプラットフォームで動作します。

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

アダプターの依存関係

- Telcordia Service Delivery Module 9.5.1、10.0、11.0 が必要です。
- コネクターが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するために使用できるソフトウェアは MQSeries 5.1、5.2[2]、および WebSphere MQ 5.3 です。そのため、これらのいずれかのソフトウェア・リリースをインストールしている必要があります。
- IBM WebSphere MQ Java クライアント・ライブラリー (WebSphere MQ 5.3 に付属) が必要です。
- XML Schema Object Discovery Agent (ODA) が必要です。このコンポーネントは、XML スキーマ文書からビジネス・オブジェクト定義を生成するときに不可欠です。この XML ODA は、アダプターをインストールするときにはインストールされません。XML ODA のインストールは、別途行う必要があります。

ロケール依存データ

コネクターは、2 バイト文字セットをサポートし、指定された言語でメッセージ・テキストを送達できるように国際化されています。ある文字コードを使用するロケーションから別の文字コード・セットを使用するロケーションへデータを転送する場合、コネクターは、そのデータの意味が伝わ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 (単一バイトおよびマルチバイトの両方) の文字エンコード方式が組み込まれています。IBM WebSphere Business Integration システムのほとんどのコンポーネントは Java で作成されています。したがって、たいていの統合コンポーネント間のデータ転送の場合には、文字変換の必要はありません。

エラーおよび通知メッセージを国または地域に応じて適切な言語でログに記録するには、ご使用の環境に応じて `Locale` 標準構成プロパティーを構成します。構成プロパティーの詳細については、57 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。

Telcordia アプリケーション・コネクターの概要

Telcordia アプリケーションは、さまざまなネットワーク・アクセス機能を介して、テレフォニー・サービス、ビデオ・サービス、およびデータ・サービスをカスタマーに提供します。Telcordia Service Delivery/Order Manager は、カスタマー・オーダーを処理します。カスタマー・オーダーはオーダー番号とオーダー・タイプにより一意的に識別されます。Telcordia 用のコネクターを使用することで、Telcordia Service Delivery Module と IBM 統合ブローカーの間での双方向データ交換が可能になります。

Telcordia Service Delivery/Order Manager には、次の 2 つの主要コンポーネントがあります。

2. ご使用の環境で文字セット変換に convert-on-the-get 方法を実装している場合は、最新の MA88 (JMS クラス) を IBM からダウンロードしてください。パッチ・レベルは 5.2.2 以上です (WebSphere MQ バージョン 5.2 の場合)。これにより、サポートされないエンコード・エラーを回避できます。

1. Service Delivery/Order Manager プロセッサー。これには、以下のものが含まれます。
   - ワークフローを処理する汎用エンジン。
   - パートナー・システム (IBM WebSphere 統合ブローカーおよび Telcordia 用のコネクターが該当) と通信するアプリケーション・タスク・ハンドラー。
   - Service Delivery/Order Manager の SRDB (Service Request Database) を更新する状況タスク・ハンドラー。
2. Order Inquiry GUI。これは、Telcordia の NGN-OSS (Next Generation Network-Operations Support Systems) および Tier 2 Inter-Domain Network Management Solution の GUI (Solution GUI とも呼ばれています) のコンポーネントです。

Telcordia 用のコネクターでは、Telcordia の Order Inquiry を使用して、カスタマー・オーダー、オーダー要求、およびオーダー項目の状況をモニターします。Order Inquiry を使用することで、以下の情報を表示できます。

- オーダー要求 (カスタマー・オーダーのコンポーネント) が処理されるときに発生するアクティビティー (処理ステップ) の状況。
- 特定のオーダー要求に関する詳細な情報
- Service Delivery/Order Manager と IBM WebSphere の間において、カスタマー・オーダー、オーダー要求、またはオーダー項目のレベルで交換されるメッセージ。

図 1 に、Telcordia Service Delivery/Order Manager と Telcordia 用 IBM WebSphere コネクターのアーキテクチャーを示します。

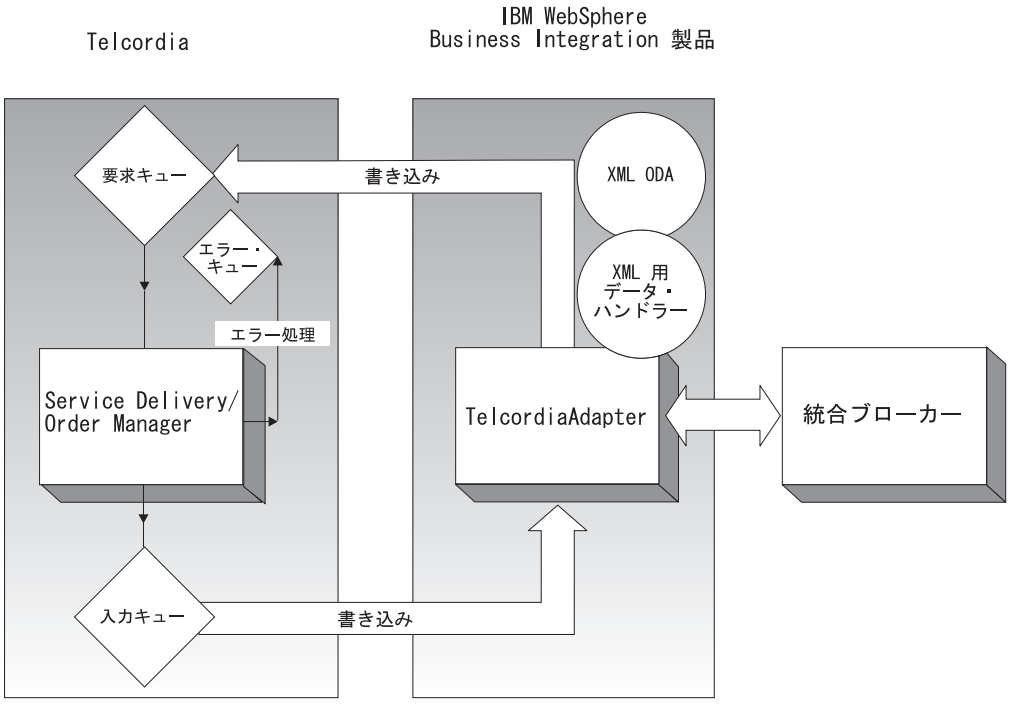

図 1. Telcordia 向け IBM WebSphere Business Integration のアーキテクチャー

図 1 は、以下のことを示しています。

- コネクターは、XML を使用して、WebSphere MQ キュー経由で Telcordia Service Delivery/Order Manager と通信します。
- IBM WebSphere Business Integration の XML 用データ・ハンドラーは、XML 文書と IBM WebSphere ビジネス・オブジェクトを相互に変換します。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。
- XML ODA は、XML スキーマ文書からビジネス・オブジェクト定義を生成するときに使用する Object Discovery Agent (ODA) です。また、XML ODA は、IBM WebSphere Business Object Designer と共に使用することもできます。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。
- IBM WebSphere 統合ブローカーは、複数の異種ビジネス・アプリケーションを 1 つのアプリケーションとして管理できる、プロセス自動化サーバーです。詳細については、「システム管理ガイド」、「*WebSphere MQ Integrator Broker 用インプリメンテーション・ガイド*」、または「*アダプター実装ガイド (WebSphere Application Server)*」を参照してください。
- 入力キュー、要求キュー、およびエラー・キューは、それぞれイベント・ポーリング、要求処理、およびエラー処理のために Telcordia サーバー上に構成されている、ローカルの WebSphere MQ キューです。入力キューには DELV_2_IBM という名前が付けられており、要求キューには IBM_2_DELV という名前が付けられています。
- Service Delivery/Order Manager では、Order Inquiry GUI を使用して、オーダー状況をモニターすることができます。

Service Delivery/Order Manager は、新規接続や切断などのタイプのカスタマー・オーダーをサポートしています。カスタマー・オーダーの処理方法はオーダー・タイプと要求されたアクションによって異なります。Telcordia 用のコネクターは、さまざまなタイプのメッセージをサポートしています。Telcordia 用のコネクターでは、以下の 6 つのタイプの Service Delivery メッセージが、テスト済みかつ使用可能なメッセージです。

- 汎用オーダー要求
- 汎用オーダー応答
- LSR 要求
- LSR 応答
- 請求オーダー完了
- 請求オーダー完了応答

Service Delivery/Order Manager は、それ自体が備えている柔軟な実行シーケンス・フロー制御機能を通じて、オーダー要求の処理を制御し、積極的にトラッキングします。処理のフローには、IBM WebSphere、Telcordia Service Delivery、Network Configuration Manager (NetCon)、および Work Item Manager (WIM) のほか、3 つのマルチステップの基本メッセージ交換が関係します。

コネクターのアーキテクチャー

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。

InterChange Server を統合ブローカーとして使用するようにコネクターが構成されている場合、IBM WebSphere Business Integration Collaborations と、Telcordia メッセージを送受信するアプリケーションとの間で、データの変更が発生したときに非同期的にビジネス・オブジェクトを交換することができます。

コネクターは、キューから Telcordia メッセージを取得し、XML 用データ・ハンドラーを呼び出してその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・オブジェクトにそのビジネス・オブジェクトをデリバリーします。また、逆に、コラボレーション・オブジェクトからビジネス・オブジェクトを取得し、同じデータ・ハンドラーを使用してそのビジネス・オブジェクトを Telcordia メッセージに変換し、Telcordia のキューにそのメッセージをデリバリーします。

コネクターには、2 つのサンプル・ビジネス・オブジェクト定義スキーマと、XML メッセージが付属しています。XML Object Discovery Agent (ODA) では、これらのスキーマやメッセージを変更したり、新しいスキーマやメッセージを作成したりできます。詳細については、49 ページの『第 3 章 ビジネス・オブジェクトの作成と変更』を参照してください。

デフォルトでは、コネクターは XML 用データ・ハンドラーを使用します。詳細については、「データ・ハンドラー・ガイド」を参照してください。

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

コネクターは、複数の入力キューをポーリングし、各キューをラウンドロビン方式でポーリングして指定された数のメッセージを各キューから検索することができます。コネクターは、ポーリング中に検索された各メッセージに、動的子メタオブジェクト (ビジネス・オブジェクトで指定されている場合) を追加します。子メタオブ

ジェクトの値は、コネクターに対し、メッセージのフォーマットおよびメッセージが検索された入力キューの名前を属性に取り込むように指示できます。

入力キューからメッセージが検索されると、コネクターは、その入力キューと、メッセージ・ヘッダーに含まれる FORMAT フィールドに関連付けられているビジネス・オブジェクト名を調べます。次に、メッセージ本体が、該当するビジネス・オブジェクトの新しいインスタンスと共に、データ・ハンドラーに渡されます。入力キューとフォーマットに関連付けられたビジネス・オブジェクト名が見つからない場合は、メッセージ本体だけがデータ・ハンドラーに渡されます。ビジネス・オブジェクトにメッセージの内容が正常に取り込まれると、コネクターはそのビジネス・オブジェクトがサブスクライブされているかどうかをチェックしてから、gotApplEvents() メソッドを使用して InterChange Server にデリバリーします。

アプリケーションとコネクターの間の通信方法

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 の目的は、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で送受信できるようにすることです。

メッセージ要求

図 2 に、メッセージ要求通信を示します。コラボレーション・オブジェクトから doVerbFor() メソッドに IBM WebSphere InterChange Server ビジネス・オブジェクトが渡されると、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。そこで、JMS レイヤーが、キュー・セッションを開いてメッセージを転送するための適切な呼び出しを行います。

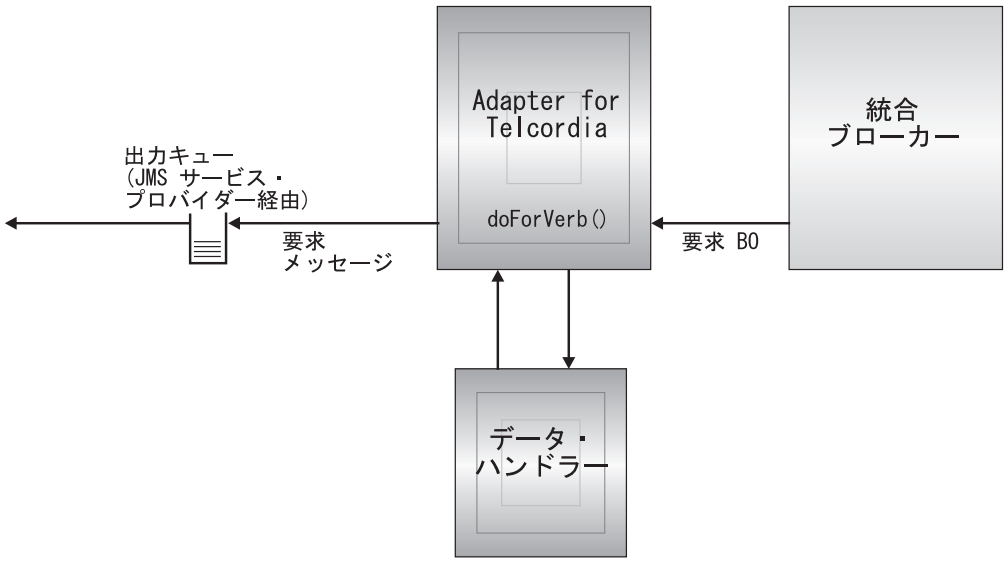

図 2. アプリケーションとコネクターの通信方式: メッセージ要求

イベント・デリバリー

図 3 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、進行中キューにステージされ、処理が完了するまでそこにとどまります。コネクターは最初に、静的メタオブジェクトまたは動的メタオブジェクトのいずれかを使用して、そのメッセージ・タイプがサポートされているかどうかを調べます。サポートされている場合、コネクターは構成されているデータ・ハンドラーにメッセージを渡し、データ・ハンドラーがそのメッセージをビジネス・オブジェクトに変換します。設定されている動詞は、メッセージ・タイプについて設定されている変換プロパティーを反映します。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・オブジェクト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は、gotApplEvents() メソッドがビジネス・オブジェクトを InterChange Server に引き渡して、そのメッセージは処理中キューから除去されます。

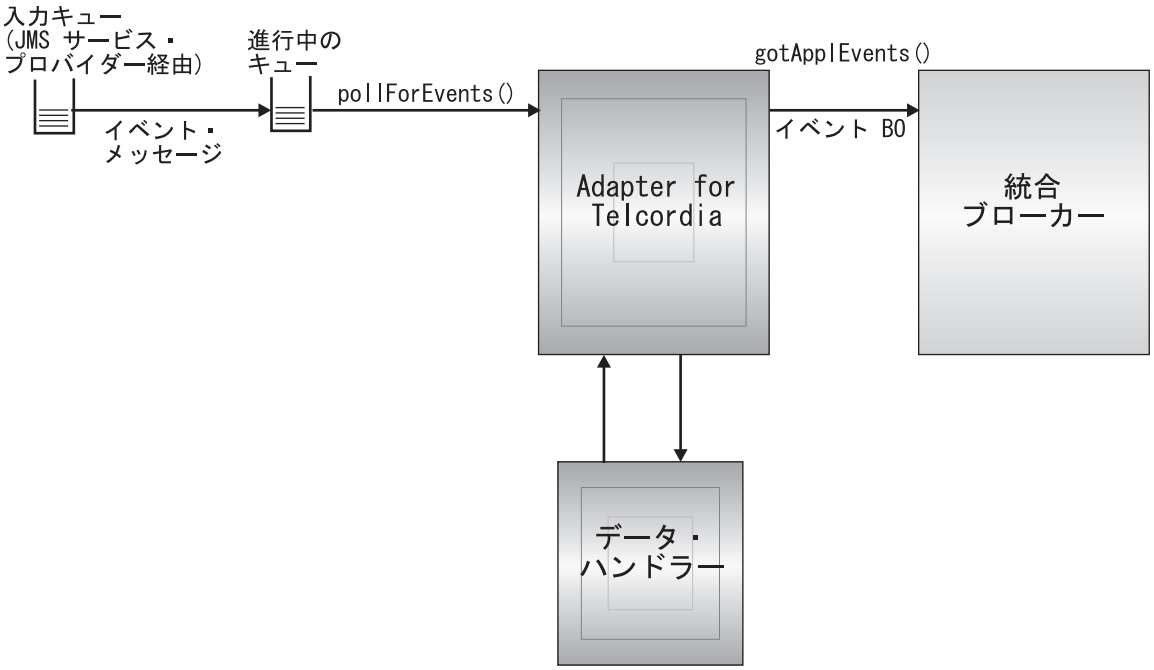

図 3. アプリケーションとコネクターの通信方式: イベント・デリバリー

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。イベントが発生するのは、Telcordia Service Delivery アプリケーションでメッセージが生成され、そのメッセージが Telcordia メッセージ・キューに格納されたときです。

検索

コネクターは、pollForEvents() メソッドを使用して Telcordia 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 Telcordia 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。イベントの失敗のシナリオについては、50 ページの『エラー処理』を参照してください。

コネクターがメッセージを処理するときには、最初に入力キューへのトランザクション・セッションが開始されます。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・オブジェクトにビジネス・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進行中キューに移動します。進行中キューでは、メッセージは処理が完了するまで保持されます。処理中にコネクターの予期しないシャットダウンが発生した場合、進行中キュー内のメッセージは、元の入力キューに復元されることなくそのまま保持されます。

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、キュー内の要求されたアクションがすべて実行され、キューからイベントが除去される前にコミットされる必要があります。したがって、コネクターは、キューからメッセージを検索する際に以下の 3 つのことが発生するまでは検索にコミットしません。1) メッセージがビジネス・オブジェクトに変換される。2) ビジネス・オブジェクトが gotApplEvents() メソッドによって InterChange Server にデリバリーされる。3) 戻り値が受信される。

同期イベント処理

Connector for Telcordia では、要求が処理された後、その結果の詳細を示すレポート・メッセージを発行してアプリケーションに返送することもできます。これにより、発行した要求に対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できます。

この処理を実現するために、コネクターはこのような要求のビジネス・データを InterChange Server に同期的に通知します。コラボレーション・オブジェクトがビジネス・オブジェクトを正常に処理した場合、コネクターは、InterChange Server からの戻りコードとビジネス・オブジェクトのすべての変更を含むレポートを要求発行者に返送します。コネクターまたはコラボレーション・オブジェクトがビジネス・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、Telcordia 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: Telcordia 対応コネクターが肯定確認通知レポートまたは否定確認通知レポート (PAN または NAN) を要求するメッセージを受け取った場合、コネクターはそのメッセージの内容を InterChange Server に同期的に通知し、レポート・メッセージに戻りコードと変更されたビジネス・データを組み込んで、要求を発行したアプリケーションに返送します。

次の表に、コネクターに送信された Telcordia 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 (複数値のサポートには論理 OR 演算子を使用する)
MessageType	メッセージ・タイプ	DATAGRAM
report	要求されたレポート・メッセージのオプション	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。
メッセージ本文		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・オブジェクト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に指定されたコラボレーション・オブジェクト名を検索します。
3. 指定されたコラボレーション・オブジェクトに、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 (複数値のサポートには論理 OR 演算子を使用する)
MessageType	メッセージ・タイプ	REPORT
feedback	レポートのタイプ	次のいずれかです。 • MQRO_PAN コラボレーション・オブジェクトがビジネス・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・オブジェクトがエラーを検出した場合に、レポートが返送されます。
メッセージ本文		コラボレーション・オブジェクトがビジネス・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・オブジェクトから戻されたビジネス・オブジェクトを取り込みます。静的メタデータ・オブジェクト内の DoNotReportBusObj プロパティーを true に設定すると、このデフォルトの振る舞いを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・オブジェクト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

コネクターの初期化時には、コネクターのシャットダウンなどが原因で完全に処理されなかったメッセージが進行中キュー内にあるかどうかが検査されます。コネクター構成プロパティー InDoubtEvents を使用すると、このようなメッセージのリカバリー処理のための 4 つのオプション (fail on startup、reprocess、ignore、または log error) のうちの 1 つを指定できます。

Fail on startup

Fail on Startup オプションを使用すると、コネクターの初期化時に進行中キュー内のメッセージが検出された場合、エラー・ログは記録されますが、コネクターは即時にシャットダウンします。メッセージを調べて適切な処置を行う (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) のは、ユーザーまたはシステム管理者の役割です。

Reprocess

Reprocess オプションを使用すると、コネクターの初期化時に進行中キュー内のメッセージが検出された場合、以降のポーリング中にこれらのメッセージが最初に処理されます。コネクターは、進行中キュー内のメッセージをすべて処理した後で、入力キューからのメッセージの処理を開始します。

Ignore

Ignore オプションを使用すると、コネクターの初期化時に進行中キュー内のメッセージが検出された場合、これらのメッセージは無視されますが、コネクターはシャットダウンしません。

Log error

Log error オプションを使用すると、コネクターの初期化時に進行中キュー内のメッセージが検出された場合、エラー・ログが記録されますが、コネクターはシャットダウンしません。

アーカイブ

コネクター・プロパティー ArchiveQueue が指定され、有効なキューが識別されている場合、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 が定義されていない場合は、メッセージは処理後に廃棄されます。アンサブスクライブされたメッセージまたはエラーが発生したメッセージのアーカイブの詳細については、49 ページの『第 3 章 ビジネス・オブジェクトの作成と変更』に記載されている 50 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、取得したメッセージを即時に別のキューに発行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、フォーマット・フィールドが、アーカイブまたは再引き渡しされるメッセージについてコピーされる唯一の追加メッセージ・プロパティーです。

保証付きイベント・デリバリー

保証付きイベント・デリバリー機能を使用すると、コネクター・フレームワークで、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 キューの間でイベントが失われたり、2 回送信されたりしないようにすることができます。JMS に対応させるには、コネクターの DeliveryTransport 標準プロパティーを JMS に構成する必要があります。このように構成すると、コネクターは JMS トランスポートを使用するため、コネクターと統合ブローカー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 トランスポートでは、最終的にメッセージが宛先に移送されることが保証されます。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場合は、コネクターがイベントをパブリッシュしたとき (コネクターがその pollForEvents() メソッド内部から gotApplEvent() メソッドを呼び出したとき) から、コネクターがイベント・レコードを削除してイベント・ストアを更新 (あるいはイベント・ストアを「event posted」状況で更新) するまでの間に、ウィンドウが表示されるとたちまち障害が発生する可能性があります。この期間に障害が発生すると、イベントは送信されますが、イベント・レコードは「進行中」状況でイベント・ストア内に残ります。コネクターは再開時に、まだイベント・ストア内にあるこのイベント・レコードを見つけて送信するため、イベントが二重送信される結果になります。

JMS イベント・ストアを有する、または有していない JMS 対応コネクターで保証付きイベント・デリバリー機能を構成することができます。保証付きイベント・デリバリーを使用するためのコネクター構成については、「*コネクター開発ガイド (Java 用)*」を参照してください。

コネクター・フレームワークがビジネス・オブジェクトを ICS 統合ブローカーに送信できない場合、オブジェクトは UnsubscribedQueue や ErrorQueue ではなく FaultQueue に置かれ、状況表示と問題の説明が生成されます。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

InterChange Server がビジネス・オブジェクトを `doVerbFor()` メソッドに送信すると、ビジネス・オブジェクト要求が処理されます。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 Telcordia メッセージに変換し、発行します。データ・ハンドラーの要件を除き、処理対象のビジネス・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・オブジェクト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。コネクターはビジネス・オブジェクト・ハンドラーと `doForVerb()` メソッドを使用して、コネクターがサポートするビジネス・オブジェクトを処理します。

注: コネクターは、ビジネス・オブジェクト動詞 Create をサポートしています。Create 動詞を持つビジネス・オブジェクトは、非同期的にも、同期的にも発行することができます。デフォルト・モードは非同期です。

Create

Create 動詞を持つビジネス・オブジェクトの処理は、そのオブジェクトの発行が同期か非同期かによって異なります。

非同期の引き渡し

これは、Create 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。メッセージは、データ・ハンドラーを使用してビジネス・オブジェクトから作成され、出力キューに書き込まれます。メッセージがデリバリーされると、コネクターは SUCCESS または FAIL を戻します。

注: コネクターには、メッセージが受信されたかどうか、あるいはアクションが実行されたかどうかを確認する手段はありません。

同期の引き渡し

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。コネクターはその後で応答を待機して、受信アプリケーションによって適切なアクションが実行されることを確認します。

Telcordia 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	変換プロパティーで定義されている出力フォーマット。IBM 要件に従い 8 文字に切り捨てられます (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	応答メッセージが必要な場合は、このフィールドにコネクター・プロパティー ReplyToQueue の値を指定します。
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。メッセージの本文は、データ・ハンドラーを使用して直列化されたビジネス・オブジェクトです。

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

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	変換プロパティーで定義されている busObj の入力フォーマット。
MessageType	メッセージ・タイプ	MQMT_REPORT*

* は、IBM によって定義される定数を示します。

動詞	フィードバック・フィールド	メッセージ本文
Create	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

Telcordia のフィードバック・コード	InterChange Server の同等な応答*
MQFB_NONE (フィードバック・コードが指定されない場合のデフォルトです)	VALCHANGE
MQFB_PAN または MQFB_APPL_FIRST	SUCCESS
MQFB_NAN または MQFB_APPL_FIRST + 1	FAIL
MQFB_APPL_FIRST + 2	VALCHANGE
MQFB_APPL_FIRST + 3	VALDUPES
MQFB_APPL_FIRST + 4	MULTIPLE_HITS
MQFB_APPL_FIRST + 5	FAIL_RETRIEVE_BY_CONTENT
MQFB_APPL_FIRST + 6	BO_DOES_NOT_EXIST
MQFB_APPL_FIRST + 7	UNABLE_TO_LOGIN
MQFB_APPL_FIRST + 8	APP_RESPONSE_TIMEOUT (その結果、即時にコネクターが終了します)

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

ビジネス・オブジェクトが処理できた場合、アプリケーションはフィードバック・フィールドを MQFB_PAN (または特定の IBM WebSphere InterChange Server の値) に設定して、レポート・メッセージを作成します。また、オプションで、すべての変更を含む直列化されたビジネス・オブジェクトをメッセージ本文に取り込みます。ビジネス・オブジェクトを処理できない場合、アプリケーションは、フィードバック・フィールドを MQFB_NAN (または特定の IBM WebSphere InterChange Server 値) に設定したレポート・メッセージを作成し、必要に応じてメッセージ本体にエラー・メッセージを組み込みます。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、メッセージの本文にビジネス・オブジェクトとエラー・メッセージのどちらが含まれているかを予測します。フィードバック・コードが MQFB_NONE (フィードバック・コードが指定されない場合のデフォルト値) の場合、コネクターはデフォルトで戻りコードが VALCHANGE であると見なします。ビジネス・オブジェクトが含まれていると予測したにもかかわらず、メッセージの本体にビジネス・オブジェクトが取り込まれていなかった場合、コネクターは InterChange Server が Request 操作のために最初に発行したのと同じビジネス・オブジェクトを単純に返送します。予期されたエラー・メッセージがメッセージ本体にない場合、応答コードと共に汎用のエラー・メッセージが InterChange Server に戻されます。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する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。これは、同期要求処理のみに適用されます (下記参照)。

メッセージ・セレクター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 が定義されていない場合は、コネクターは、上記の相関 ID を使用して応答メッセージを識別します。

response_selector が定義されている場合は、名前と値の組が以下の構文で格納されている必要があります。

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

メッセージ・セレクター・ストリングは応答を一意的に識別しなければならず、値は以下のように単一引用符で囲まれていなければなりません。

```
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 correlationID を持つ応答メッセージの ReplyToQueue をモニターします。アダプターはこのメッセージ・セレクター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。例えば、以下のメッセージ・セレクターでは、

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

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 の場合は、アダプターは以下の基準で作成されたメッセージ・セレクターを生成し、使用して

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

応答メッセージを識別します。

以下のように複数の置換を指定することもできます。

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

この例では、アダプターは {1} をトップレベル・ビジネス・オブジェクトの属性 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 を使用した深い検索の方法については、JCDK API のマニュアル (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。

- 「{}」 シンボルの間に非整数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、リテラル値「{」または「}」をメッセージ・セレクターに含める場合は、それぞれ「{{」または「{}」を使用してください。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・セレクターを以下のように解決します。

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

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

属性値からリテラル・ストリング置換を抽出する方法を次の例に示します。

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

MyDynamicMO.CorrelationID に値 {A:B}C;D が含まれていると、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 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、Telcordia 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、InterChange Server 固有の戻り状況値のすべてが Telcordia フィードバック・コードにマップされているメタオブジェクトを作成することができます。(メタオブジェクトを使用して) フィードバック・コードに割り当てられた戻り状況値は、InterChange Server に渡されます。詳細については、28 ページの『FeedbackCodeMappingMO』を参照してください。

共通の構成タスク

インストール後、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いくつかの構成タスクと始動タスクについて概説します。

アダプターのインストール

何をどこへインストールする必要があるかについては、23 ページの『第 2 章 コネクターのインストールと構成』を参照してください。

コネクター・プロパティーの構成

コネクターの構成プロパティーには、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という 2 つのタイプがあります。これらのプロパティーには、変更する必要のないデフォルト値を持つものもあります。コネクターを実行する前に、これらのプロパティーの一部の値を設定する必要があります。詳細については、23 ページの『第 2 章 コネクターのインストールと構成』を参照してください。

Adapter for Telcordia のコネクター・プロパティーを構成するときには、以下の条件を満たしている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、Telcordia サーバーのホストの値と一致している。
- コネクター・プロパティー Port に指定した値が、キュー・マネージャーのリスナーのポートと一致している。
- コネクター・プロパティー Channel に指定した値が、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、 UnsubscribeQueue のキュー URI が有効で、実際に存在する。
- CCSID プロパティーの値が、Telcordia サーバー上の WebSphere MQ キュー・マネージャーの CCSID 値と一致している。Windows ベースのコネクターの場合、CCSID 値は 819 です。

通知なしで要求を送信するようにコネクターを構成

通知なしで要求を送信するように (「発信後削除」とも呼ばれるデフォルトの非同期モード) コネクターを構成するには、次の手順を実行します。

- 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。
- 静的または動的いずれかのメタオブジェクトを使用して、ターゲット・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、38 ページの『静的メタオブジェクト作成の概要』と 40 ページの『動的子メタオブジェクト作成の概要』を参照してください。
- メタオブジェクト (静的または動的) のプロパティー ResponseTimeout を -1 に設定します。この場合、コネクターはビジネス・オブジェクトを発行しますが、戻り値を確認することはありません。
- 詳細については、13 ページの『Create』、 33 ページの『メタオブジェクトの構成』、および 49 ページの『第 3 章 ビジネス・オブジェクトの作成と変更』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、 18 ページの『通知なしで要求を送信するようにコネクターを構成』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、13 ページの『Create』を参照してください。応答メッセージが、ここに挙げられている要件を満たさないと、コネクターは、エラーを報告したり、異常終了して応答メッセージを認識できなかったりすることがあります。 33 ページの『メタオブジェクトの構成』と 49 ページの『第 3 章 ビジネス・オブジェクトの作成と変更』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には、ビジネス・オブジェクトと、コネクターがビジネス・オブジェクトを処理する方法について指定した、アプリケーション固有の情報が収められています。コネクターは、始動時に、静的メタオブジェクトからビジネス・オブジェクトを処理するために必要な情報をすべて取得します。

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、構成するには、次のようにします。

- 38 ページの『静的メタオブジェクト作成の概要』の手順に従います。
- コネクター固有のプロパティー ConfigurationMetaObject 内で静的メタオブジェクトの名前を指定することにより、コネクターが静的メタオブジェクトにサブスクライブするようにします。詳細については、25 ページの『コネクター固有のプロパティー』を参照してください。
- デフォルト・プロパティーを使用する場合は、CCSID が URI の一部として指定されていることを確認します。Windows ベースのコネクターを実行する場合、CCSID の値を 819 に設定します。

動的メタオブジェクトの構成

コネクターが、シナリオに応じてビジネス・オブジェクトの処理方法を変更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により、コネクターは (ランタイムで) 要求を処理する方法を認識します。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情報をすべて提供する静的メタオブジェクトと異なり、動的メタオブジェクトが提供するのは、特定のシナリオで要求を処理する場合に必要なロジックの追加要素だけです。動的メタオブジェクトを作成、構成するには、次のようにします。

- 動的メタオブジェクトを作成し、子として要求ビジネス・オブジェクトに追加します。
- コネクターに発行する前に動的メタオブジェクトにターゲット・キュー、メッセージ・フォーマットなどの情報を取り込む追加ロジックを、コラボレーション・オブジェクトに組み込みます。

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

デフォルト・プロパティーを使用する場合は、CCSID が URI の一部として指定されていることを確認します。Windows ベースのコネクターを実行する場合、CCSID の値を 819 に設定します。

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、アプリケーション間でメッセージをやり取りする場合に、アプリケーション・データと一緒に渡される制御情報が収められています。静的または動的いずれかのメタオブジェクトで、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、13 ページの『Create』を参照してください。

キュー URI の構成

Adapter for Telcordia 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、以下のとおりです。

  `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 ヘッダーのみを処理し、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、キュー URI に `?targetClient=1` を付加します。詳細については、31 ページの『キュー 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には、次の手順に従います。

- コネクター固有のプロパティー DataHandlerClassName に、データ・ハンドラー・クラスの名前 (`com.crossworlds.DataHandlers.text.xml`) を指定します。詳細については、25 ページの『コネクター固有のプロパティー』を参照してください。
- XML データ・ハンドラー・メタオブジェクト (`MO_DataHandler_DefaultXMLConfig`) 内に、データ・ハンドラー・クラスの名前 (`com.crossworlds.DataHandlers.text.xml`) を指定します。
- コネクター固有のプロパティー DataHandlerMimeType と DataHandlerConfigMO に、MIME タイプと、その MIME タイプの構成が定義されているデータ・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。
- データ・ハンドラー・メタオブジェクト内に、ネーム・ハンドラーの完全なクラス・パスを指定します。ネーム・ハンドラー・クラス (TopElementNameHandler) は、XML メッセージからルート・エレメント名を抽出するときに使用されま

す。NameHandlerClass プロパティーの値は、
com.crossworlds.DataHandlers.xml.TopElementNameHandler になります。

始動スクリプトの変更

コネクターの始動方法の詳細については、 23 ページの『第 2 章 コネクターのインストールと構成』を参照してください。始動前にコネクターのプロパティーを構成する必要があります。また、始動ファイルを変更する必要があります。

- クライアント・ライブラリーのロケーションを指すように、start_connector スクリプトを必ず変更してください。複数のバージョン、または現在使用している Telcordia サーバーに対応しないバージョンのクライアント・ライブラリーをインストールしないようにしてください。詳細については、44 ページの『始動ファイルの構成』を参照してください。

第 2 章 コネクターのインストールと構成

この章では、コネクターのインストールと構成の方法、コネクターと連動するようにメッセージ・フローを構成する方法、およびコネクターの開始と停止の方法について説明します。

インストール作業の概要

アダプターをインストールするには、以下の手順を実行します。

アダプターの前提条件の確認

アダプターをインストールする前に、アダプターをインストールおよび実行するための環境の前提条件がご使用のシステムですべて満たされていることを確認してください。詳細については、1 ページの『アダプター環境』を参照してください。

統合ブローカーのインストール

統合ブローカーのインストールでは、WebSphere Business Integration システムのインストールとブローカーの始動を実行します。この作業については、ご使用のブローカーの資料を参照してください。Connector for Telcordia がサポートするブローカーの詳細については、2 ページの『ブローカーの互換性』 を参照してください。

ブローカーのインストールついての詳細は、ご使用のブローカーのインプリメンテーション・ガイドを参照してください。

Adapter for Telcordia と関連ファイルのインストール

WebSphere Business Integration アダプター製品のインストールについては、次のサイトで WebSphere Business Integration Adapters Infocenter にある「*WebSphere Business Integration Adapters インストール・ガイド*」を参照してください。

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

インストール済みファイルの構造

コネクター・コンポーネントは、既存のインストールの追加コンポーネントとしてインストールします。これを行うには、IBM WebSphere Business Integration Adapter のインストーラーを実行し、IBM WebSphere Business Integration Adapter for Telcordia を選択します。

Windows のコネクター・ファイル構造

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

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

以下の表に、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、インストーラーを使用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。

ProductDir のサブディレクトリー	説明
connectors¥Telcordia¥CWTelcordia.jar	Telcordia コネクターが使用するクラスを含みます。
connectors¥Telcordia¥start_Telcordia.bat	コネクターの始動スクリプト (Windows 2000)。
connectors¥Telcordia¥Repository¥TelcordiaConnectorTemplate	アダプター定義のテンプレート・ファイル。
repository¥Telcordia¥CN_Telcordia.txt	コネクターのリポジトリー定義。

注: すべての製品パス名は、システム上の、製品のインストール先ディレクトリーを基準とした相対パス名です。

UNIX コネクター・ファイル構造

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

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

以下の表に、コネクターが使用する UNIX ファイル構造が記載されており、インストーラーを使用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。

ProductDir のサブディレクトリー	説明
connectors/Telcordia/CWTelcordia.jar	Telcordia コネクターが使用するクラスを含みます。
connectors/Telcordia/start_Telcordia.sh	コネクターのシステム始動スクリプト。このスクリプトは、汎用のコネクター・マネージャー・スクリプトから呼び出されます。System Manager の「コネクター構成」画面をクリックすると、インストーラーは、このコネクター・マネージャー・スクリプト用にカスタマイズされたラッパーを作成します。コネクターの始動および停止には、このカスタマイズ済みラッパーを使用してください。

ProductDir のサブディレクトリー	説明
connectors/messages/TelcordiaConnector.txt	コネクターのメッセージ・ファイル。
repository/Telcordia/CN_Telcordia.txt	コネクターのリポジトリー定義。

注: すべての製品パス名は、システム上の、製品のインストール先ディレクトリーを基準とした相対パス名です。

コネクターの構成

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

Connector Configurator を使用してコネクター・プロパティーを構成します。

- Connector Configurator の説明と手順については、77 ページの『付録 B. Connector configurator』を参照してください。
- 標準コネクター・プロパティーの説明については、『標準コネクター・プロパティー』、および 57 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説明については、『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 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、57 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

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

コネクター固有のプロパティー

コネクター固有の構成プロパティーは、コネクターが実行時に必要とする情報を提供します。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。

以下の表に、コネクターのアダプ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*		はい
ApplicationUserName	*ログイン・ユーザー ID*		はい
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	なし
CCSID	*キュー・マネージャーの接続に使用する文字セット*		なし
Channel	*Telcordia サーバー・コネクター・チャネル*	CHANNEL1	はい
ConfigurationMetaObject	*構成メタオブジェクトの名前*	Telcordia_MQSeries_MO_Config	はい
DataHandlerClassName	データ・ハンドラー・クラス名	com.crossworlds.DataHandlers. text.xml	なし
DataHandlerConfigMO	データ・ハンドラー・メタオブジェクト	MO_DataHandler_ Default	はい
DataHandlerMimeType	ファイルの *MIME* タイプ	Text/XML	はい
DefaultVerb	コネクターがサポートしている任意の動詞。		
EnableMessageProducerCache	true または false	true	なし
ErrorQueue	*未処理のメッセージ・キュー*	queue://crossworlds.queue.manager/MQCONN.ERROR	なし
FeedbackCodeMappingMO	フィードバック・コード・メタオブジェクト		なし
HostName	*Telcordia サーバー*		はい
InDoubtEvents	FailOnStartup Reprocess IgnoreLogError	Reprocess	なし
InputQueue	ポーリング・キュー	queue://crossworlds.queue.manager/DELV_2_IBM	なし
InProgressQueue	*進行中のイベント・キュー*	queue://crossworlds.queue.manager/MQCONN.IN_PROGRESS	なし
PollQuantity	*InputQueue プロパティーに指定された各キューから検索するメッセージ数*	1	なし
Port	*WebSphere MQ リスナー用に設定されたポート*	1414	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デリバリーされるキュー*	queue://crossworlds.queue.manager/MQCONN.REPLYTO	なし
SessionPoolSizeForRequests	*要求処理中に使用されるセッションをキャッシングするための最大プール・サイズ*	10	なし
UnsubscribedQueue	*アンサブスクライブされたメッセージが送信されるキュー*	queue://crossworlds.queue.manager/MQCONN.UNSUBSCRIBED	なし
UseDefaults	true または false	false	

ApplicationPassword

Telcordia にログインする場合に UserID と併用するパスワード。

デフォルト = 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 Telcordia 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

Telcordia にログインする場合に Password と併用するユーザー ID。

デフォルト = 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 Telcordia 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の送信先のキュー。

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

CCSID

キュー・マネージャー接続の文字セット。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。31 ページの『キュー Uniform Resource Identifiers (URI)』を参照してください。この値は、Windows ベースの Telcordia 用のコネクターの場合、819 に設定する必要があります。

デフォルト = なし。

Channel

コネクターが Telcordia との通信に使用する MQ サーバー・コネクター・チャネルです。

デフォルト = CHANNEL1。

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 Telcordia 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が収められている静的メタオブジェクトの名前。

デフォルト = Telcordia_MQSeries_MO_Config。

DataHandlerClassName

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

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

DataHandlerConfigMO

構成情報を提供するためにデータ・ハンドラーに渡されるメタオブジェクトです。

デフォルト = MO_DataHandler_Default

DataHandlerMimeType

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

デフォルト = Text/XML

DefaultVerb

ポーリングのときにデータ・ハンドラーによって着信ビジネス・オブジェクト内に動詞が設定されていない場合に設定すべき動詞を指定します。

デフォルト = なし

EnableMessageProducerCache

要求メッセージを送信するために、アダプターがメッセージ・プロジューサーのキャッシュを有効にすることを指定する boolean プロパティー。

Default= true

ErrorQueue

処理されなかったメッセージが送信されるキューです。

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

FeedbackCodeMappingMO

InterChange Server にメッセージの受信を同期で肯定応答する場合に使用す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、再割り当てすることができます。このプロパティーを使用すれば、各属性名がフィードバック・コードを表すメタオブジェクトを指定できます。フィードバック・コードの対応する値は、InterChange Server に渡される戻り状況です。デフォルト・フィードバック・コードのリストについては、14 ページの『同期の引き渡し』を参照してください。コネクターは、Telcordia 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*。ここで *N* は整数を表します *(*MQFB_APPL_FIRST+ *N* の値と解釈されます*)*。

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

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

表 1. 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 = なし。

HostName

Telcordia のホストとなるサーバーの名前です。

デフォルト = なし。

InDoubtEvents

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

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

デフォルト = Reprocess。

InputQueue

コネクターが新規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、MyQueueC の 3 つのキューをポーリングする場合、コネクター構成プロパティー *InputQueue* の値は MyQueueA;MyQueueB;MyQueueC になります。

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。

pollQuanity の値は 2 のため、コネクターは pollForEvents への呼び出しを行うたびに各キューから多くても 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 pollQuanity が 2 に設定されているため、コネクターは第 2 ラウンドのポーリングを開始し、MyQueueA と MyQueueC の各キューからそれぞれ 1 つずつのメッセージを検索します。このとき、MqQueueB は空になっているためスキップされます。すべてのキューをそれぞれ 2 回ずつポーリングした後、メソッド pollForEvents への呼び出しは完了します。メッセージを取り出す順序は以下のようになります。

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

デフォルト = queue://crossworlds.queue.manager/DELV_2_IBM

InProgressQueue

処理中にメッセージを保持するメッセージ・キュー。System Manager を使用してコネクター固有のプロパティーからデフォルトの InProgressQueue 名を削除することによって、このキューなしで動作するようコネクターを構成できます。そのように設定すると、イベントの保留中にコネクターがシャットダウンされたときにイベント・デリバリーが影響を受ける場合があるという警告のプロンプトが、始動時に出されます。

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

PollQuantity

pollForEvents スキャン時に InputQueue プロパティーで指定された各キューから取得するメッセージの数。

デフォルト = 1

Port

MQ リスナー用に確立するポートです。

デフォルト = 1414

ReplyToQueue

コネクターからの要求発行時に応答メッセージがデリバリーされるキューです。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、42 ページの『JMS ヘッダ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

SessionPoolSizeForRequests

要求処理中に使用されるセッションをキャッシングするための最大プール・サイズ。

Default = 10

UnsubscribedQueue

アンサブスクライブされたメッセージが送信されるキューです。

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

注: * Telcordia 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。問題がある場合は、暗黙的に値を指定してください。

UseDefaults

Create 操作では、UseDefaults が true に設定されている場合、コネクターは、isRequired となっている各ビジネス・オブジェクト属性に有効な値またはデフォルト値が指定されているかをチェックします。値が指定されていれば、Create 操作は正常に実行されます。このパラメーターが false に設定されている場合、コネクターは有効な値についてのチェックしか行わないため、有効な値が指定されていないと Create 操作は失敗します。デフォルト値は false です。

キュー Uniform Resource Identifiers (URI)

キューの URI は、先頭が queue:// で、その後に以下の要素が続きます。

- キューが常駐するキュー・マネージャーの名前
- 別の /
- キューの名前
- 必要に応じて、その他のキュー・プロパティーを設定する name-value 形式のリスト

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 DELV_2_IBM 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/DELV_2_IBM?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。正整数は、タイムアウト (ミリ秒単位) を表します。
priority	メッセージの優先順位。	0 から 9 で、1 が最も高い優先順位です。値 -1 は、キューの構成によってプロパティーが決まることを意味します。値 -2 は、コネクターが独自のデフォルト値を使用できることを指定します。

プロパティー名	説明	値
persistence	メッセージをディスクに「永久保存」するかどうかを指定します。	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、コネクターが独自のデフォルト値を使用できることを指定します。
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。27 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 に準拠しているかどうかを指定します。	0 = JMS (MQRFH2 ヘッダー) 1 = MQ (MQMD ヘッダーのみ)
encoding	数値フィールドを表す方法を指定します。	基本 WebSphere MQ マニュアルに記載されている整数値です。

注: コネクターは、MQMessage 内のデータの文字セット (CCSID) またはエンコード属性を制御できません。データ変換はデータがメッセージ・バッファーから検索されるかメッセージ・バッファーにデリバリーされるときに行われるため、コネクターは 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 やエンコードのメッセージを転送するには、出力キューが完全修飾 URI であることと、CCSID と encoding の値を指定することが必要です。コネクターはこの情報を WebSphere MQ に渡し、WebSphere MQ は MQMessage のデリバリーのためにデータをエンコードするときに (JMS API を介して) この情報を使用します。CCSID やエンコードがサポートされていない場合は、通常、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で解決できます。CCSID やエンコードに固有の問題が引き続き発生する場合には、IBM WebSphere InterChange Server テクニカル・サポートに連絡して、代替 Java 仮想マシンを使用することでコネクターを実行できるかどうかを問い合わせてください。

メタオブジェクトの構成

コネクターは、メタオブジェクト・エントリーを使用して、メッセージと関連付けるビジネス・オブジェクトを決定します。イベント・メッセージの処理に使用されるビジネス・オブジェクトのタイプと動詞は、 WebSphere MQ メッセージ・ヘッダーに含まれる FORMAT フィールドに基づいて決定されます。ビジネス・オブジェクト名と動詞を格納するメタオブジェクト属性を構成し、WebSphere MQ メッセージ・ヘッダーの FORMAT フィールド・テキストに関連付けます。メタオブジェクト属性には、メッセージ処理のガイドラインも格納されます。

入力キューからメッセージを検索する場合、コネクターは、FORMAT テキスト・フィールドに対応するビジネス・オブジェクト名を探します。次に、ビジネス・オブジェクトの名前とともに、メッセージがデータ・ハンドラーに渡されます。ビジネス・オブジェクトにメッセージの内容が正常に取り込まれると、コネクターはそのビジネス・オブジェクトがサブスクライブされているかどうかをチェックしてから、gotApplEvents() メソッドを使用して統合ブローカーにデリバリーします。

コネクターは、2 種類のメタオブジェクトを認識し、読み取ることができます。

- 静的コネクター・メタオブジェクト
- 動的子メタオブジェクト

動的子メタオブジェクトの属性値は、静的メタオブジェクトの属性値をコピーしてオーバーライドします。

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

- **静的メタオブジェクト**
  - 各メッセージに対するメタデータがすべて決まっていて、構成時に指定できる場合に有効です。
  - ビジネス・オブジェクト・タイプによって値を指定するよう制限されます。例えば、Customer タイプ・オブジェクトはすべて同じ宛先に送信する必要があります。
- **動的メタオブジェクト**
  - ビジネス・プロセスがメッセージ・ヘッダーの情報にアクセスできるようにします。
  - ビジネス・タイプに関係なく、ビジネス・プロセスが、実行時にメッセージの処理を変更できるようにします。例えば、動的メタオブジェクトを使用すると、アダプターに送られる各 Customer タイプ・オブジェクトに別々の宛先を指定できます。
  - サポートされるビジネス・オブジェクトの構造を変更する必要があります。— そのような変更では、マップおよびビジネス・プロセスを変更しなければならない場合があります。
  - カスタム・データ・ハンドラーを変更する必要があります。

メタオブジェクト・プロパティー

表 2 は、メタオブジェクトでサポートされるプロパティーの完全なリストです。メタオブジェクトをインプリメントする場合は、これらのプロパティーを参照してください。メタオブジェクトには、表 2 に示された 1 つ以上のプロパティーが必要です。

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

表 2. Telcordia アダプター・メタオブジェクトのプロパティー

プロパティー名	静的メタオブジェクトで定義可能	動的メタオブジェクトで定義可能	説明
CollaborationName	はい	はい	CollaborationName は、ビジネス・オブジェクトと動詞の組み合わせに対する属性のアプリケーション固有のテキスト内で指定される必要があります。例えば、ユーザーが動詞 Create 付きのビジネス・オブジェクト Customer の同期イベント・デリバリーを処理することを想定している場合、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、 38 ページの『静的メタオブジェクト作成の概要』 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。

表 2. *Telcordia アダプター・メタオブジェクトのプロパティー (続き)*

プロパティー名	静的メタオブジェクトで定義可能	動的メタオブジェクトで定義可能	説明
DataHandlerConfigMO	はい	はい	構成情報を提供するためにデータ・ハンドラーに渡されるメタオブジェクトです。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る場合は、DataHandlerConfigMO コネクター・プロパティーで指定されている値がオーバーライドされます。このメタオブジェクト・プロパティーは、さまざまなタイプのビジネス・オブジェクトを処理するために複数の異なるデータ・ハンドラーが必要な場合に使用します。データ形式が実際のビジネス・データに依存する可能性がある場合は、要求処理には動的な子メタオブジェクトを使用します。指定されたビジネス・オブジェクトはコネクター・エージェントによりサポートされていることが必要です。 77 ページの『付録 B. Connector configurator』 の説明を参照してください。
DataHandlerMimeType	はい	はい	使用すると、特定の MIME タイプに基づいたデータ・ハンドラーを要求できます。メタオブジェクトに指定されている場合、この値は DataHandlerMimeType コネクター・プロパティーに指定されている値をオーバーライドします。このメタオブジェクト・プロパティーは、さまざまなタイプのビジネス・オブジェクトを処理するために複数の異なるデータ・ハンドラーが必要な場合に使用します。データ形式が実際のビジネス・データに依存する可能性がある場合は、要求処理には動的な子メタオブジェクトを使用します。DataHandlerConfigMO に指定されたビジネス・オブジェクトは、このプロパティーの値に対応する属性を含める必要があります。 77 ページの『付録 B. Connector configurator』 の説明を参照してください。
DataHandlerClassName	はい	はい	77 ページの『付録 B. Connector configurator』 の説明を参照してください。
InputFormat	はい	はい	所定のビジネス・オブジェクトに関連付けるインバウンドの (イベント) メッセージのフォーマットまたはタイプです。この値は、メッセージを生成したアプリケーションによって指定され、メッセージの内容を識別するのに役立ちます。検索されたメッセージがこのフォーマットである場合、そのメッセージは (可能であれば) 特定のビジネス・オブジェクトに変換されます。ビジネス・オブジェクトにこのフォーマットが指定されていない場合、コネクターは特定のビジネス・オブジェクトのサブスクリプション・デリバリーを処理しません。このプロパティーを設定するときは、デフォルトのメタオブジェクト変換プロパティーを使用しないでください。このプロパティーの値は、着信メッセージをビジネス・オブジェクトと突き合わせるために使用されます。メッセージ・フォーマットの定義でコネクターが考慮するフィールドは、コネクター固有プロパティー MessageFormatProperty によって、ユーザーが定義できます。

表 2. Telcordia アダプター・メタオブジェクトのプロパティー (続き)

プロパティー名	静的メタオブジェクトで定義可能	動的メタオブジェクトで定義可能	説明
OutputFormat	はい	はい	アウトバウンド・メッセージで読み込まれるフォーマット。OutputFormat が指定されていない場合は、入力フォーマットが使用されます (使用可能な場合)。
InputQueue	はい	はい	コネクターが新規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です。このプロパティーは、着信メッセージをビジネス・オブジェクトとマッチングする目的のみで使用されます。このプロパティーを設定するときは、デフォルトの変換プロパティーを使用しないでください。デフォルトの変換プロパティーの値は、着信メッセージをビジネス・オブジェクトに一致させるために使用されます。 **注:** コネクター固有のプロパティーとしての InputQueue プロパティーは、アダプターのポーリング先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、アダプターが所定のメッセージを特定のビジネス・オブジェクトにマップする基準としての役割を果たします。この機能をインプリメントするためには、コネクター固有のプロパティーを使用して複数の入力宛先を構成し、オプションで、着信メッセージの入力フォーマットに基づき異なるデータ・ハンドラーをそれぞれにマップします。詳細については、 39 ページの『データ・ハンドラーの入力キューへのマッピングの概要』を参照してください。
OutputQueue	はい	はい	所定のビジネス・オブジェクトから派生したメッセージを引き渡す出力キューです。
ResponseTimeout	はい	はい	同期要求処理の応答を待機した状態で、タイムアウトになるまでの時間をミリ秒で表します。このプロパティーが定義されないままかまたはゼロより小さい値が設定されている場合、コネクターは応答が戻るまで待機せずにすぐに SUCCESS を戻します。
TimeoutFatal	はい	はい	同期要求処理において、応答が受信されない場合にコネクターにエラー・メッセージを戻させるために使用されます。このプロパティーが True の場合、応答が ResponseTimeout で指定した時間内に受信されなければ、コネクターはブローカーに APPRESPONSETIMEOUT を戻します。このプロパティーが未定義か、または False にセットされている場合、応答がタイムアウトになるとコネクターは要求をエラーにしますが、終了はしません。デフォルト = False。

表 2. *Telcordia アダプター・メタオブジェクトのプロパティー (続き)*

プロパティー名	静的メタオブジェクトで定義可能	動的メタオブジェクトで定義可能	説明
DataEncoding			DataEncoding は、メッセージの読み取り/書き込みに使用するエンコードです。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の読み取りを試みます。動的子メタオブジェクトで定義されている DataEncoding は、静的メタオブジェクトで定義されている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。
JMS メッセージ・ヘッダーに限定してマッピングされるフィールドを以下に示します。説明、値の解釈などについては、JMS API 仕様を参照してください。JMS プロバイダーは、一部のフィールドについて、異なった解釈をする場合があります。それらの違いについて、JMS プロバイダーの資料も確認してください。			
ReplyToQueue		はい	要求に対する応答メッセージの送信先となるキューです。
Type		はい	メッセージのタイプ。JMS プロバイダーによって異なりますが、一般には、ユーザーによる定義が可能。
MessageID		はい	メッセージの固有 ID (JMS プロバイダーに固有)。
CorrelationID	はい	はい	この応答を開始した要求メッセージの ID を示すために、応答メッセージで使用されます。
Delivery Mode	はい	はい	MOM システムでメッセージを永続させるかどうかを指定します。許容値は以下のとおりです。 1 = 非永続 2 = 永続 JMS プロバイダーによっては、その他の値を使用できます。
Priority		はい	メッセージの優先順位 (数値)。許容値は、0 から 9 です (数値が小さいほど優先順位は高くなる)。
Destination		はい	MOM システムでの、メッセージの現在または最後の (削除された場合) 場所。
Expiration		はい	メッセージの存続時間。
Redelivered		はい	JMS プロバイダーが以前にクライアントへのメッセージ配信を試みた可能性は高いが、受取が確認されなかったことを示します。
Timestamp		はい	時間メッセージが JMS プロバイダーに渡されました。
UserID		はい	メッセージを送信するユーザーの ID。
AppID		はい	メッセージを送信するアプリケーションの ID。
DeliveryCount		はい	配信を試みる回数。
GroupID		はい	メッセージ・グループの ID。
GroupSeq		はい	グループ ID で指定されたメッセージ・グループにおける、このメッセージの順序。
JMSProperties		はい	42 ページの『JMS プロパティー』を参照してください。

静的メタオブジェクト作成の概要

Telcordia アダプター構成のメタオブジェクトは、さまざまなビジネス・オブジェクト用に定義された変換プロパティーのリストで構成されます。静的メタオブジェクトのサンプルを表示するには、Business Object Designer を起動し、アダプターに同梱された以下のサンプルを開きます。

connectors¥WebSpereMQ¥samples¥LegacyContact¥WebSpereMQ_MO_Config.xsd

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

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

表 3. 静的メタオブジェクトの構造

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

以下のメタオブジェクトを例にとります。

表 4. 静的メタオブジェクト構造のサンプル

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

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

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

せが、それらの個別のメタデータ・プロパティー以外に、
OutputDestination=QueueA を持つとみなします。

表 5. 静的メタオブジェクト構造のサンプル

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

34 ページの表 2 には、静的メタオブジェクト内でアプリケーション固有の情報として指定できるプロパティーが示されています。

注: 静的メタオブジェクトが指定されていない場合、コネクターはポーリング時にメッセージ・フォーマットを特定のビジネス・オブジェクト・タイプにマップできません。このような場合、コネクターはビジネス・オブジェクトを指定せずに、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・ハンドラーがこのテキストのみではビジネス・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できないことを示すエラーをレポートします。

静的メタオブジェクトの作成手順

静的メタオブジェクトをインプリメントするには、以下を実行します。

1. Business Object Designer を起動します。詳細については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。
2. 34 ページの表 2 を参照して、要件に適合するように属性および ASI を指定してから、メタオブジェクト・ファイルを保管します。
3. このメタオブジェクト・ファイルの名前を、コネクター・プロパティー ConfigurationMetaObject の値として指定します。

データ・ハンドラーの入力キューへのマッピングの概要

静的メタオブジェクトのアプリケーション固有の情報に InputQueue プロパティーを使用すれば、データ・ハンドラーを入力キューに関連付けることができます。この機能は、フォーマットや変換要件の異なる取引先が複数ある場合に有効です。

データ・ハンドラーを入力キューにマップする手順

データ・ハンドラーを InputQueue にマップするには、以下を行います。

1. コネクター固有のプロパティー (29 ページの『InputQueue』 を参照) を使用して、1 つ以上の入力キューを構成します。
2. **Business Object Designer** で静的メタオブジェクトを開きます。
3. 静的メタオブジェクトの入力キューごとに、アプリケーション固有の情報として、キュー・マネージャー、入力キュー名、データ・ハンドラー・クラス名、MIME タイプを指定します。

例えば、以下に示す静的メタオブジェクトの属性は、CompReceipts という名前の InputQueue にデータ・ハンドラー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.WBIMB.disposition_notification;DataHandlerMimeType=
message/
disposition_notification
IsRequiredServerBound = false
[End]
```

動的子メタオブジェクト作成の概要

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

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

コネクターは、コネクターに渡されるトップレベル・ビジネス・オブジェクトに子として追加される動的なメタオブジェクトから、変換プロパティーを認識し、読み取ります。コネクターを構成する場合に使用する静的メタオブジェクトで指定できる変換プロパティーは、動的子メタオブジェクトの属性値にコピーされます。

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。

34 ページの表 2 には、動的メタオブジェクト内でアプリケーション固有の情報として指定できるプロパティーが示されています。

以下の属性は JMS 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 6. 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

表 6. 動的メタオブジェクト・ヘッダー属性 (続き)

ヘッダー属性名	モード	対応する JMS ヘッダー
Type	読み取り	JMSType
UserID	読み取り	JMSXUserID
AppID	読み取り	JMSXAppID
DeliveryCount	読み取り	JMSXDeliveryCount
GroupID	読み取り	JMSXGroupID
GroupSeq	読み取り	JMSXGroupSeq
JMSProperties	読み取り/書き込み	

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

動的メタオブジェクトは、各属性がそれぞれ 1 つのメタデータ・プロパティーと値を meta-object property name =meta-object property value の形式で表す構造になっています。

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

ポーリング時の動的子メタオブジェクトの取り込み

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

表 7 に、動的な子メタオブジェクトがポーリング用に構造化される方法を示します。

表 7. ポーリング用の動的子メタオブジェクトの構造

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

表 7 に示すように、動的子メタオブジェクトで追加の属性 Input_Format および InputQueue を定義できます。Input_Format は検索したメッセージのフォーマットで読み込まれ、InputQueue 属性には特定のメッセージの検索先となる宛先の名前が含まれます。子メタオブジェクト内にこれらのプロパティーが定義されていない場合、これらには値が取り込まれません。

シナリオ例:

- コネクターは、キュー MyInputQueue から、フォーマットが CUST_IN の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チェックして、メタオブジェクトが定義されているかどうか判別します。
- メタオブジェクトが定義されている場合、コネクターはこのメタオブジェクトのインスタンスを作成し、定義に基づいて InputQueue および InputFormat 属性に値を取り込んで、ビジネス・オブジェクトを使用可能なコラボレーションにパブリッシュします。

JMS ヘッダーおよび動的子メタオブジェクト属性

メッセージ・トランスポートに関して、より多くの情報を取得したり、制御の範囲を広げるための属性を動的メタオブジェクトに追加することができます。このセクションでは、これらの属性、およびそれらがイベント通知と要求処理にどのような影響を与えるかについて説明します。

JMS プロパティー: 動的メタオブジェクトの他の属性とは異なり、JMSProperties には単一カーディナリティーの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内のすべての属性には、JMS メッセージ・ヘッダーの変数部分で読み/書きされる単一プロパティーを以下のように定義します。

1. 属性の名前は、セマンティック値ではありません。
2. 属性のタイプは、JMS プロパティー・タイプにかかわらず、常に String です。
3. 属性のアプリケーション固有の情報には、この属性がマップする JMS メッセージ・プロパティーの名前とフォーマットを定義した 2 組の名前と値のペアを記述します。名前はユーザー定義可能です。値の型は、以下のいずれかでなければなりません。
   - Boolean
   - String
   - Int
   - Float
   - Double
   - Long
   - Short
   - Byte

次の表は、JMSProperties オブジェクトの属性として定義する必要のあるアプリケーション固有の情報のプロパティーを示します。

表 8. JMS プロパティー属性のアプリケーション固有の情報

属性	指定可能な値	ASI	コメント
名前	有効な JMS プロパティー名 (有効 = ASI で定義されたタイプと互換性がある)	name=<*JMS プロパティー名*>;type=<*JMS プロパティー・タイプ*>	ベンダーによっては、拡張機能を提供するために特定のプロパティーが予約されていることがあります。このようなベンダー固有の機能にアクセスしようとする場合を除き、一般にユーザーは JMS で始まるカスタム・プロパティーを定義しないでください。
Type	String	type=<*コメントを参照*>	これは JMS プロパティーのタイプです。JMS API には、JMS Message に値を設定するメソッドとして setIntProperty、setLongProperty、setStringProperty などがあります。ここで指定する JMS プロパティーのタイプは、これらのうちのどのメソッドを使用してメッセージにプロパティー値を設定するかを示します。

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

```
Customer (ASI = cw_mo_conn=MetaData)
  |-- Id
  |-- FirstName
  |-- LastName
  |-- ContactInfo
  |-- MetaData
        |-- OutputFormat = CUST
        |-- OutputDestination = QueueA
        |-- JMSProperties
              |-- RoutingCode = 123 (ASI= name=RoutingCode;type=Int)
              |-- Dept = FD (ASI= name=RoutingDept;type=String)
```

別の例を示すために、図 4 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。この属性のアプリケーション固有の情報には、それぞれの名前とタイプが定義されています。例えば属性 ID は、String タイプの JMS プロパティー ID と対応しています。

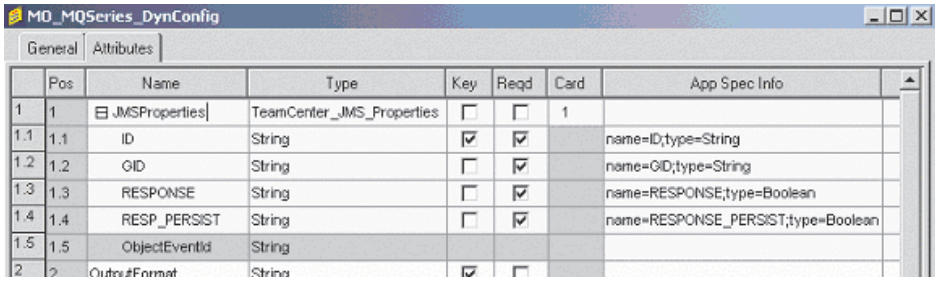

図 4. 動的メタオブジェクトの JMS プロパティー属性

動的メタオブジェクトの作成手順

動的メタオブジェクトをインプリメントするには、以下を実行します。

1. **Business Object Designer** を起動します。詳細については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。
2. 動的メタオブジェクトに対する処理を操作したい、トップレベルのビジネス・オブジェクトを開きます。
3. 動的メタデータ・オブジェクトを子としてトップレベル・オブジェクトに追加し、トップレベル・オブジェクト ASI に名前と値のペア `cw_mo_conn=<MO attribute>` を組み込みます。*<MO attribute>* は、動的メタオブジェクトを表すトップレベル・オブジェクトの属性名です。例えば、次のようになります。

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

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

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

始動ファイルの構成

Connector for Telcordia を始動する前に、始動ファイルを構成する必要があります。

Windows

Windows のコネクターの構成を完了するには、`start_Telcordia.bat` ファイルを修正する必要があります。

1. `start_Telcordia.bat` ファイルを開きます。

2. 「Set the directory containing your WebSphere MQ Java client libraries」で始まるセクションにスクロールし、使用する WebSphere MQ Java クライアント・ライブラリーのロケーションを指定します。

UNIX

UNIX プラットフォーム用のコネクターを構成するには、以下の手順に従って start_Telcordia.sh ファイルを変更する必要があります。

1. start_Telcordia.sh ファイルを開きます。
2. 「Set the directory containing your WebSphere MQ Java client libraries」で始まるセクションにスクロールし、使用する WebSphere MQ Java クライアント・ライブラリーのロケーションを指定します。

複数のコネクター・インスタンスの作成

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

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

新規ディレクトリーの作成

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

```
ProductDir¥connectors¥connectorInstance
```

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

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

```
ProductDir¥repository¥connectorInstance
```

ビジネス・オブジェクト定義の作成

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

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

ProductDir¥repository¥initialConnectorInstance

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

コネクター定義の作成

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

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

始動スクリプトの作成

始動スクリプトは以下のように作成します。

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

   dirname
2. この始動スクリプトを、45 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. 始動スクリプトのショートカットを作成します (Windows のみ)。
4. 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。

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

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

コネクターの始動

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

ProductDir¥connectors¥*connName*

ここで、*connName* はコネクターを示します。始動スクリプトの名前は、表 9 に示すように、オペレーティング・システム・プラットフォームによって異なります。

表 9. コネクターの始動スクリプト

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

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

- Windows システムで**「スタート」**メニューから。

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

    `start_connName connName brokerName [-cconfigFile ]`
  - UNIX ベースのシステム:

    `connector_manager_connName -start`

  ここで、*connName* はコネクターの名前であり、*brokerName* は以下のようにご使用の統合ブローカーを表します。
  - WebSphere InterChange Server の場合は、*brokerName* に ICS インスタンスの名前を指定します。
  - WebSphere Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、または WebSphere Business Integration Message Broker) または WebSphere Application Server の場合は、*brokerName* にブローカーを示すストリングを指定します。

  注: Windows システム上の WebSphere Message Broker または WebSphere Application Server の場合は、`-c` オプションに続いてコネクター構成ファイルの名前を指定しなければなりません。ICS の場合は、`-c` はオプションです。
- Adapter Monitor から (WebSphere Business Integration Adapters 製品のみ)。Adapter Monitor は System Manager 始動時に起動されます。

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

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

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

- WebSphere InterChange Server については、「*システム管理ガイド*」を参照してください。

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

コネクターの停止

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

- コマンド行からコネクターを始動した場合は、コネクター始動スクリプトを用いて、以下の操作を実行します。
  - Windows システムでは、始動スクリプトを起動すると、そのコネクター用の別個の「コンソール」ウィンドウが作成されます。このウィンドウで、「Q」と入力して Enter キーを押すと、コネクターが停止します。
  - UNIX ベースのシステムでは、コネクターはバックグラウンドで実行されるため、別ウィンドウはありません。代わりに、次のコマンドを実行してコネクターを停止します。

    ```
    connector_manager_connName -stop
    ```

    ここで、*connName* はコネクターの名前です。
- Adapter Monitor から (WebSphere Business Integration Adapters 製品のみ)。Adapter Monitor は System Manager 始動時に起動されます。

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

  このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。
- Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムのシャットダウン時に、コネクターは停止します。

第 3 章 ビジネス・オブジェクトの作成と変更

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。IBM WebSphere InterChange Server ビジネス・オブジェクトでは、メタデータはアプリケーションに関するデータです。このデータは、ビジネス・オブジェクト定義内に保管されており、コネクターとアプリケーションの相互作用を支援します。メタデータ主導型コネクターは、サポートする各ビジネス・オブジェクトを、コネクター内にハードコーディングされた命令ではなく、ビジネス・オブジェクト定義にエンコードされたメタデータに基づいて処理します。

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、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なので、コネクター・コードを変更しなくても、新しいビジネス・オブジェクトまたは変更されたビジネス・オブジェクトを処理することができます。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、Telcordia 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。

コネクター・ビジネス・オブジェクトの構造

コネクターをインストールした後は、ビジネス・オブジェクトを作成するか、コネクターに付属しているビジネス・オブジェクトを使用 (または変更) する必要があります。

Telcordia 用のコネクターでは、Telcordia 用のスキーマ・ファイルを処理することができます。各スキーマ・ファイルには、対応する IBM WebSphere InterChange Server ビジネス・オブジェクトがあります。これらのビジネス・オブジェクトの生成には、XML Object Discovery Agent (ODA) を使用します。これらのオブジェクトには、`Telcordia_` などの接頭部を指定する必要があります。命名規則に従い、

TelcordiaPrefix_SchemaFileType の形式の名前にしてください。テスト済みのアダプター固有のビジネス・オブジェクトは、次の 6 つです。

- Telcordia_OrderRequest
- Telcordia_OrderResponse
- Telcordia_LSRRequest
- Telcordia_LSRResponse
- Telcordia_BillingOrderCompletionResponse
- Telcordia_BillingOrdercompletionContract

XML ODA を使用したビジネス・オブジェクトの生成の詳細については、52 ページの『スキーマ文書に基づく Telcordia ビジネス・オブジェクトの生成』を参照してください。

ビジネス・オブジェクトの構造に関する要件については、構成済みのデータ・ハンドラーとの関係で必要となる要件以外の要件はありません。命名規則の詳細については、「*コンポーネント命名ガイド*」を参照してください。

コネクターはキューからメッセージを検索し、ビジネス・オブジェクト (メタオブジェクトによって定義されたもの) にメッセージの内容を取り込もうとします。厳密にいうと、コネクターがビジネス・オブジェクト構造を制御したり、この構造に影響を及ぼしたりすることはありません。ビジネス・オブジェクト構造を規定する機能を果たすのは、メタオブジェクト定義とコネクターのデータ・ハンドラーの要件です。実際、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存在しません。ビジネス・オブジェクトを検索して渡す際のコネクターの主な役割は、メッセージとビジネス・オブジェクト間のプロセスでエラーをモニターすることです。

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、`TelcordiaConnector.txt` という名前のメッセージ・ファイルに格納されます。(ファイル名は、`LogFileName` 標準コネクター構成プロパティーによって決定されます)。各エラーには、エラー番号とそれに続くエラー・メッセージが含まれます。

```
Message number
Message text
```

コネクターは、以下の各セクションで説明するような特定のエラーを処理します。

適切に形成されていない要求と無効な要求

処理不能な要求への対処方法には、次の 2 つがあります。

- Telcordia Service Delivery がポーリングした要求キュー (IBM_2_DELV) のメッセージが、適切に形成されておらず処理不能である場合、そのメッセージは定義済みのエラー・キュー DELV_ERROR_QUEUE に送られます。ICS 統合ブローカーは、応答を受け取りません。
- Telcordia Service Delivery は、コネクターから無効な要求を受け取った場合、次のような <MSG> (メッセージ) セクションを含む応答メッセージを生成します。

<MSG> <CODE> 0173 </CODE> <TEXT> LSO NOT FOUND IN WC MAPING REFERENCE TABLE </TEXT> <ERRTAGPATH>
DELV.SR.RU.DATA.CKT.NEWACL.LSO </ERRTAGPATH> </MSG>

このような応答やオーダー状態の変更が発生した場合には、コラボレーション・オブジェクトが ICS 統合ブローカーに戻るようにする必要があります。

アプリケーション・タイムアウト

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

- メッセージの検索中は、コネクターは JMS サービス・プロバイダーへの接続を確立できません。
- 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、接続が確立されていないため、発信キューにそれを渡すことができません。
- 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` に設定されたビジネス・オブジェクトの応答を待機していてタイムアウトになってしまいます。
- コネクターは、APP_RESPONSE_TIMEOUT または UNABLE_TO_LOGIN に等しい戻りコードを持つ応答メッセージを受け取りました。

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廃棄されます。

`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 キュー内に残ります。

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をビジネス・オブジェクトに変換できなかった場合や (JMS プロバイダーではなく) ビジネス・オブジェクトに固有の処理エラーが発生した場合、メッセージは、`ErrorQueue` で指定されたキューに送信されます。`ErrorQueue` が定義されていない場合、エラーが原因で処理できないメッセージは廃棄されます。

データ・ハンドラーがビジネス・オブジェクトをメッセージに変換できない場合は、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あり、この機能をオンにするとコネクターの動作を密着して追跡できます。トレース・メッセージは、デフォルトでは STDOUT に書き込まれます。トレース・メッセージの構成の詳細については、23 ページの『第 2 章 コネクターのインストールと構成』に記載されている『コネクター構成プロパティー』を参照してください。トレースの有効化および設定の方法など、詳細については、「*コネクター開発ガイド*」を参照してください。

次に、コネクター・トレース・メッセージに推奨する内容を示します。

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

レベル 1　このレベルは、処理された各ビジネス・オブジェクトについて主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規メッセージを検出したときにそれを記録するトレース・メッセージに使用されます。

レベル 2　ビジネス・オブジェクトが `gotApplEvent()` または `executeCollaboration()` のいずれかから InterChange Server に通知されるごとにトレース・メッセージをログに記録する場合は、このレベルを使用します。

レベル 3　このレベルは、メッセージからビジネス・オブジェクト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送達に関する情報を提供するトレース・メッセージに使用されます。

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　このレベルは、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示すトレース・メッセージ、メッセージがキューから取り出されたりキューに入れられたりしたときにそれを記録するトレース・メッセージ、ビジネス・オブジェクトのダンプを記録するトレース・メッセージなどに使用されます。

スキーマ文書に基づく Telcordia ビジネス・オブジェクトの生成

XML スキーマ文書に基づいて要求ビジネス・オブジェクトと応答ビジネス・オブジェクトを作成する場合、処理される XML 文書のタイプごとに、ビジネス・オブジェクト定義を作成する必要があります。ビジネス・オブジェクト定義には、XML 文書のスキーマに含まれる構造情報が含まれています。例えば、1 つの要求ストリーム (1 つのスキーマ文書) があり、一方 4 つの応答ストリーム・タイプ (4 つの異なるスキーマ文書) が存在する可能性がある場合、5 つのビジネス・オブジェクト定義を作成する必要があります。これに対し、要求ストリームと応答ストリームが同じスキーマを使用する場合には、必要なビジネス・オブジェクト定義は 1 つのみです。XML Object Discovery Agent (ODA) を使用すると、スキーマ文書に基づいてビジネス・オブジェクト定義を生成することができます。

XML ODA を介してまたは手動で XML 文書用のビジネス・オブジェクト定義を生成する方法の詳細については、「データ・ハンドラー・ガイド」を参照してください。

ODA の制限

同一スキーマに組み込まれているファイルのいくつかに同様の名前のエレメント定義が含まれる場合、そのスキーマは XML ODA ではサポートされません。したがって、XML ODA では、そのようなスキーマに対して IBM WebSphere ビジネス・オブジェクト定義を正しく生成できません。この問題を回避するには、以下の処理を行います。

1. 複数の組み込みファイルで使用されており、かつそれぞれで定義が異なっている同名のエレメントを確認します。
2. それらの組み込みファイルの 1 つで、該当のエレメントの名前を、末尾に「1」を付けて変更します。その後、変更したファイルを保管します。
3. XML ODA を使用して、そのスキーマに対応する CW ビジネス・オブジェクト定義を生成し、.in ファイルに保管します。
4. 保管した .in ファイルを開き、エレメント名に付加した「1」をすべて除去して、変更したエレメント名を元の名前に戻します。
5. 生成したビジネス・オブジェクト定義ファイルを、使用するリポジトリーにロードします。これで問題なく機能します。

例えば、スキーマ・ファイル BilingOrderCompletionContract.xsd に、CRS.xsd と ORDERINP.xsd の 2 つのファイルが組み込まれているとします。また、ORDERINP.xsd には DELV.xsd が組み込まれており、このファイルにはさらに DELVTAGS.xsd が組み込まれているとします。この例では、予備手段は次のようになります。

1. 同様の名前を持つエレメント定義を確認します。ここでは、CRS.xsd と DELVTAGS.xsd の両方に、SLOT という名前のエレメントが含まれているとします。
2. CRS.xsd で SLOT を SLOT1 にリネームし、このファイルを保管します。
3. XML ODA を使用して BillingOrderCompletionContract スキーマに対応するビジネス・オブジェクト定義を生成し、生成した定義を BillingContract.in に保管します。
4. BillingContract.in をワード・プロセッシング・アプリケーションで開いて SLOT1 を SLOT で置換した後、このファイルをもう一度保管します。
5. repos_copy を使用して、BillingContract.in をリポジトリーにロードします。

注: Adapter for Telcordia リリース 2.3 を使用している場合、XML ODA はトップレベル・ビジネス・オブジェクトのキー属性を自動的に選択できません。それ以外のレベルのビジネス・オブジェクトでは、XML ODA は最初の属性をキーに設定します。したがって、Business Object Designer で XML ODA が生成したオブジェクトを保管すると、トップレベルのオブジェクトにキー属性が欠落していることを知らせるエラー・メッセージが出されます。ビジネス・データとビジネス・オブジェクトの要件を満たすキー属性を割り当ててから、オブジェクトを再度保管してください。

第 4 章 トラブルシューティング

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

- 『始動時の問題』
- 『イベント処理時の問題』

始動時の問題

問題	可能性のある解決方法/説明
初期化中にコネクターが予期せずシャットダウンし、メッセージ「Exception in thread "main" java.lang.NoClassDefFoundError: javax/jms/JMSException…」が報告される。	コネクターが IBM WebSphere MQ Java クライアント・ライブラリーからファイル jms.jar を検出できません。start_connector.bat 内の変数 MQSERIES_JAVA_LIB が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認します。
初期化中にコネクターが予期せずシャットダウンし、メッセージ「Exception in thread "main" java.lang.NoClassDefFoundError: com/ibm/mq/jms/MQConnectionFactory...」が報告される。	コネクターが IBM WebSphere MQ Java クライアント・ライブラリーからファイル com.ibm.mqjms.jar を検出できません。start_connector.bat 内の変数 MQSERIES_JAVA_LIB が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認します。
初期化中にコネクターが予期せずシャットダウンし、メッセージ「Exception in thread "main" java.lang.NoClassDefFoundError: javax/naming/Referenceable...」が報告される。	コネクターが IBM WebSphere MQ Java クライアント・ライブラリーからファイル jndi.jar を検出できません。start_connector.bat 内の変数 MQSERIES_JAVA_LIB が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認します。
初期化中にコネクターが予期せずシャットダウンし、例外「java.lang.UnsatisfiedLinkError: no mqjbnd01 in shared library path」が報告される。	コネクターが IBM WebSphere MQ Java クライアント・ライブラリーから必須のランタイム・ライブラリー (mqjbnd01.dll [NT] または libmqjbnd01.so [Solaris]) を検出できません。パスに IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーが含まれていることを確認してください。
コネクターによって「MQJMS2005: failed to create MQQueueManager for ':'」が報告される。	プロパティー HostName、Channel、および Port の値を明示的に設定します。

イベント処理時の問題

問題	可能性のある解決方法/説明
コネクターによって、MQRFH2 ヘッダーを持つすべてのメッセージがデリバリーされる。	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、23 ページの『第 2 章 コネクターのインストールと構成』を参照してください。

問題	可能性のある解決方法/説明
コネクターによって、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 文字を超える部分が切り捨てられる。	これは WebSphere MQ MQMD メッセージ・ヘッダーの制限であり、コネクターの制限ではありません。

付録 A. コネクターの標準構成プロパティー

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

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

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

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

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

新規プロパティーと削除されたプロパティー

以下の標準プロパティーは、本リリースで追加されました。

新規プロパティー

- XMLNameSpaceFormat

削除されたプロパティー

- RestartCount

標準コネクター・プロパティーの構成

Adapter 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
- コネクター固有の構成プロパティー

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

Connector Configurator の使用

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

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

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 (WebSphere InterChange Server が統合ブローカーである場合のみ)
3. ローカル構成ファイル
4. コマンド行

コネクターは、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。コネクターが System Manager から独立してスタンドアロン・モードで稼働している場合 (例えば、いずれかの WebSphere Message Brokers と連携している場合) は、構成ファイルでのみプロパティーを変更できます。この場合、動的更新は実行できません。
- **エージェント再始動 (ICS のみ)**
  アプリケーション固有のコンポーネントを停止して再始動しなければ、変更が有効になりません。
- **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。

特定のプロパティーの更新方法を確認するには、「Connector Configurator」ウィンドウ内の**「更新メソッド」**列を参照するか、次に示す 59 ページの表 10 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 10 は、標準コネクター構成プロパティーの早見表です。標準プロパティーの依存関係は RepositoryDirectory に基づいているため、コネクターによっては使用されないプロパティーがあり、使用する統合ブローカーによってプロパティーの設定が異なる可能性があります。

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

注: 表 10 の「注」列にある「Repository Directory は REMOTE」という句は、ブローカーが InterChange Server であることを示します。ブローカーが WMQI または WAS の場合には、リポジトリー・ディレクトリーは LOCAL に設定されます。

表 10. 標準構成プロパティーの要約

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

表 10. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
ControllerStoreAndForwardMode	`true` または `false`	`true`	動的	Repository Directory は <REMOTE> (ブローカーは ICS)
ControllerTraceLevel	`0` から `5`	`0`	動的	Repository Directory は <REMOTE> (ブローカーは ICS)
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	コンポーネント再始動	JMS トランスポートのみ
DeliveryTransport	`MQ`、`IDL`、または `JMS`	`JMS`	コンポーネント再始動	Repository Directory がローカルの場合は、値は JMS のみ
DuplicateEventElimination	`true` または `false`	`false`	コンポーネント再始動	JMS トランスポートのみ: Container Managed Events は <NONE> でなければならない
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	`CxCommon.Messaging.jms.IBMMQSeriesFactory` または `CxCommon.Messaging.jms.SonicMQFactory` または任意の Java クラス名	`CxCommon.Messaging.jms.IBMMQSeriesFactory`	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	FactoryClassName が IBM の場合は `crossworlds.queue.manager` を使用。FactoryClassName が Sonicの場合 `localhost:2506` を使用。	`crossworlds.queue.manager`	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	`10`	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	`128m`	コンポーネント再始動	Repository Directory は <REMOTE> (ブローカーは ICS)

表 10. 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
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> (ブローカーは ICS)
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> (ブローカーは ICS)
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならない
Locale	en_US、 ja_JP、 ko_KR、 zh_CN、 zh_TW、 fr_FR、 de_DE、 it_IT、 es_ES、 pt_BR **注:** これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	true または false	false	コンポーネント再始動	Repository Directory は <REMOTE> でなければならない (ブローカーは ICS)
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならない (ブローカーは ICS)
MessageFileName	パスまたはファイル名	CONNECTORNAMEConnector.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	*CONNECTORNAME*/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は true でなければならない
OADAutoRestartAgent	true または false	false	動的	Repository Directory は <REMOTE> でなければならない (ブローカーは ICS)

表 10. 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
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければならない (ブローカーは ICS)
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければならない (ブローカーは ICS)
PollEndTime	HH:MM	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒) no (ポーリングを使用不可にする) key (コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力された場合にのみポーリングする)	10000	動的	
PollQuantity	1 から 500	1	エージェント再始動	JMS トランスポートのみ: Container Managed Events を指定
PollStartTime	HH:MM (HH は 0 から 23、 MM は 0 から 59)	HH:MM	コンポーネント再始動	
RepositoryDirectory	メタデータ・リポジトリーの場所		エージェント再始動	ICS の場合は <REMOTE> に設定する。WebSphere MQ Message Brokers および WAS の場合: C:¥crossworlds¥repository に設定する
RequestQueue	有効な JMS キュー名	*CONNECTORNAME*/REQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
ResponseQueue	有効な JMS キュー名	*CONNECTORNAME*/RESPONSEQUEUE	コンポーネント再始動	Delivery Transport が JMS の場合: Repository Directory が <REMOTE> の場合のみ必要

表 10. 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
RestartRetryCount	0 から 99	3	動的	
RestartRetryInterval	適切な正数 (単位: 分): 1 から 2147483547	1	動的	
RHF2MessageDomain	mrm、xml	mrm	コンポーネント再始動	Delivery Transport が JMS であり、かつ WireFormat が CwXML である。
SourceQueue	有効な WebSphere MQ 名	*CONNECTORNAME*/SOURCEQUEUE	エージェント再始動	Delivery Transport が JMS であり、かつ Container Managed Events が指定されている場合のみ
SynchronousRequestQueue		*CONNECTORNAME*/SYNCHRONOUSREQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
SynchronousRequestTimeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	Delivery Transport は JMS
SynchronousResponseQueue		*CONNECTORNAME*/SYNCHRONOUSRESPONSEQUEUE	コンポーネント再始動	Delivery Transport は JMS
WireFormat	CwXML、CwBO	CwXML	エージェント再始動	Repository Directory が <REMOTE> でない場合は Repository Directory が <REMOTE> の場合は CwBO
WsifSynchronousRequestTimeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	WAS のみ
XMLNameSpaceFormat	short、long	short	エージェント再始動	WebSphere MQ Message Brokers および WAS のみ

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

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

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

RepositoryDirectory が <REMOTE> の場合のみ適用されます。

AgentConnections プロパティーは、orb.init[] により開かれる ORB (オブジェクト・リクエスト・ブローカー) 接続の数を制御します。

このプロパティーのデフォルト値は 1 に設定されます。必要に応じてこの値を変更できます。

AgentTraceLevel

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

ApplicationName

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

BrokerType

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

CharacterEncoding

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

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

デフォルトでは、ドロップダウン・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。ドロップダウン・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 に関するセクションを参照してください。

ConcurrentEventTriggeredFlows

RepositoryDirectory が <REMOTE> の場合のみ適用されます。

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

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

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

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

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

ContainerManagedEvents

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

デフォルト値はありません。

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 = /SOURCEQUEUE

また、MimeType、DHClass (データ・ハンドラー・クラス)、および DataHandlerConfigMOName (オプションのメタオブジェクト名) プロパティーを設定したデータ・ハンドラーも構成する必要があります。これらのプロパティーの値を設定するには、Connector Configurator の**「データ・ハンドラー」**タブを使用します。

これらのプロパティーはアダプター固有ですが、**例**の値は次のようになります。

- MimeType = text¥xml

- DHClass = com.crossworlds.DataHandlers.text.xml
- DataHandlerConfigMOName = MO_DataHandler_Default

「データ・ハンドラー」タブのこれらの値のフィールドは、ContainerManagedEvents を JMS に設定した場合にのみ表示されます。

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

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

ControllerStoreAndForwardMode

RepositoryDirectory が <REMOTE> の場合のみ適用されます。

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

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

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

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

デフォルト値は true です。

ControllerTraceLevel

RepositoryDirectory が <REMOTE> の場合のみ適用されます。

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

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

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

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

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

WebSphere MQ および IDL

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

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

JMS

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

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

このプロパティーは、false に設定することもできます。

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

FaultQueue

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

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

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

デフォルト値は 128M です。

JvmMaxNativeStackSize

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

デフォルト値は 128K です。

JvmMinHeapSize

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

デフォルト値は 1M です。

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

デフォルト値は crossworlds.queue.manager です。ローカル・メッセージ・ブローカーに接続する場合は、デフォルト値を使用します。

リモート・メッセージ・ブローカーに接続すると、このプロパティーは次の (必須) 値をとります。
QueueMgrName:<Channel>:<HostName>:<PortNumber>
各変数の意味は以下のとおりです。
QueueMgrName: キュー・マネージャー名です。
Channel: クライアントが使用するチャネルです。
HostName: キュー・マネージャーの配置先のマシン名です。
PortNumber: キュー・マネージャーが listen に使用するポートの番号です。

例えば、次のようになります。
jms.MessageBrokerName = WBIMB.Queue.Manager:CHANNEL1:RemoteMachine:1456

jms.NumConcurrentRequests

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

デフォルト値は 10 です。

jms.Password

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

デフォルトはありません。

jms.UserName

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

デフォルトはありません。

ListenerConcurrency

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

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

Locale

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

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、以下のように説明されます。

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

デフォルトでは、ドロップダウン・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。ドロップダウン・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 に関するセクション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

RespositoryDirectory が <REMOTE> の場合のみ適用されます。

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

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

MaxEventCapacity

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

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、製品ディレクトリーの ¥connectors¥messages です。メッセージ・ファイルが標準位置に格納されていない場合は、メッセージ・ファイル名を絶対パスで指定します。

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

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

MonitorQueue

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

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

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

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

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

デフォルト値は false です。

OADMaxNumRetry

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

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

デフォルト値は 1000 です。

OADRetryTimeInterval

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

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

デフォルト値は 10 です。

PollEndTime

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

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

PollFrequency

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

- ポーリングし、PollQuantity の値により指定される数のオブジェクトを取得します。
- これらのオブジェクトを処理します。一部のアダプターでは、これは個別のスレッドで部分的に実行されます。これにより、次のポーリング・アクションまで処理が非同期に実行されます。
- PollFrequency で指定された間隔にわたって遅延します。
- このサイクルを繰り返します。

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 (整数)。
- 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。このようなコネクターが存在する場合には、アダプターのインストールと構成に関する章で制約事項が説明されています。

PollQuantity

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。

電子メール・メッセージもイベントと見なされます。コネクターは、電子メールに関するポーリングを受けたときには次のように動作します。

コネクターは、1 回目のポーリングを受けると、メッセージの本文を選出します。これは、本文が添付とも見なされるからです。本文の MIME タイプにはデータ・ハンドラーが指定されていないので、コネクターは本文を無視します。
コネクターは PO の最初の添付を処理します。この添付の MIME タイプには対応する DH があるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
2 回目のポーリングを受けると、コネクターは PO の 2 番目の添付を処理します。この添付の MIME タイプには対応する DH があるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
これが受け入れられると、PO の 3 番目の添付が届きます。

PollStartTime

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

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

RequestQueue

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

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

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

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

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

ResponseQueue

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

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

RestartRetryCount

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

デフォルト値は `3` です。

RestartRetryInterval

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

デフォルト値は `1` です。

RHF2MessageDomain

WebSphere Message Brokers および WAS でのみ使用されます。

このプロパティーにより、JMS ヘッダーのドメイン名フィールドの値を構成できます。JMS トランスポートを介してデータを WMQI に送信するときに、アダプター・フレームワークにより JMS ヘッダー情報、ドメイン名、および固定値 `mrm` が書き込まれます。この構成可能なドメイン名により、ユーザーは WMQI ブローカーによるメッセージ・データの処理方法を追跡できます。

サンプル・ヘッダーを以下に示します。

```
<mcd><Msd>mrm</Msd><Set>3</Set><Type>
Retek_POPhyDesc</Type><Fmt>CwXML</Fmt></mcd>
```

デフォルト値は mrm ですが、このプロパティーには xml も設定できます。このプロパティーは、DeliveryTransport が JMS に設定されており、かつ WireFormat が CwXML に設定されている場合にのみ表示されます。

SourceQueue

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

JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、65 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

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

WsifSynchronousRequestTimeout

WAS 統合ブローカーでのみ使用されます。

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

デフォルト値は 0 です。

XMLNameSpaceFormat

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

XML 形式のビジネス・オブジェクト定義でユーザーがショート・ネーム・スペースとロング・ネーム・スペースを指定できる強力なプロパティー。

デフォルト値は short です。

付録 B. Connector configurator

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

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

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

注:

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

この付録では、次のトピックについて説明します。

Connector Configurator の概要

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

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

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

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

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

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

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

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

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

ただし**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクター固有プロパティーのテンプレートを作成します。システム内で既にテンプレートが作成されている場合には、作成されているテンプレートを使用します。システム内でまだテンプレートが作成されていない場合には、80 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

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

Connector Configurator の始動

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

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

スタンドアロン・モードでの Configurator の実行

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

これを行うには、以下のステップを実行します。

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

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

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (85 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

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

コネクター固有のプロパティー・テンプレートの作成

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

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

- テンプレートの新規作成については、80 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でこのテンプレートを保管します。既存のテンプレートは ¥WebSphereAdapters¥bin¥Data¥App ディレクトリーにあります。

新規テンプレートの作成

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

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

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

一般特性の指定

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

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

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

値の指定

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

1. **「値」**タブをクリックします。「一般」のパネルに代わって「値」の表示パネルが表示されます。
2. **「プロパティーを編集」**表示でプロパティーの名前を選択します。
3. **「最大長」**および**「最大複数値」**のフィールドに値を入力します。

新規プロパティー値を作成するには、以下のステップを実行します。

1. **「プロパティーを編集」**リストでプロパティーを選択し、右マウス・ボタンでクリックします。
2. ダイアログ・ボックスから**「追加」**を選択します。
3. 新規プロパティー値の名前を入力し、「OK」をクリックします。右側の**「値」**パネルに値が表示されます。

「値」パネルには、3 つの列からなるテーブルが表示されます。

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

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

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

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

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

依存関係の設定

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

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

1. **「使用可能なプロパティー」**表示で、依存プロパティーとして指定するプロパティーを選択します。

2. **「プロパティーを選択」**フィールドで、ドロップダウン・メニューを使用して、条件値を持たせるプロパティーを選択します。
3. **「条件演算子」**フィールドで以下のいずれかを選択します。

   == (等しい)

   != (等しくない)

   > (より大)

   < (より小)

   >= (より大か等しい)

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

新しい構成ファイルの作成

構成ファイルを新規に作成するには、構成ファイルの名前を指定し、統合ブローカーを選択する必要があります。

- 「System Manager」ウィンドウで**「コネクター」**フォルダーを右クリックし、**「新規コネクターの作成」**を選択します。Connector Configurator が開き、**「新規コネクター」**ダイアログ・ボックスが表示されます。
- スタンドアロン・モードの場合は、Connector Configurator で**「ファイル」>「新規」>「コネクター構成」**を選択します。「新規コネクター」ウィンドウで、新規コネクターの名前を入力します。

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

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

コネクター固有のテンプレートからの構成ファイルの作成

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

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

- **名前**

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

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

- **システム接続**

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

- **「コネクター固有プロパティー・テンプレート」を選択します。**

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

  使用するテンプレートを選択し、**「OK」**をクリックします。

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

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

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

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

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

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

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

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

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

1. Connector Configurator 内で、**「ファイル」>「開く」>「ファイルから」**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)
   - ICS リポジトリー (*.in、*.out)

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

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

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

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

構成ファイルの完成

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

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

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

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

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

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

構成ファイル・プロパティーの設定

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

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

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

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

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

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

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

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

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

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

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

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下のステップを実行します。

1. 値を設定するフィールド内でクリックします。

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

アプリケーション固有の構成プロパティーの設定

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

1. グリッドの左上端の部分で右マウス・ボタンをクリックします。ポップアップ・メニュー・バーが表示されます。プロパティーを追加するときは**「追加」**をクリックします。子プロパティーを追加するには、親の行番号で右マウス・ボタンをクリックし、**「子を追加」**をクリックします。
2. プロパティーまたは子プロパティーの値を入力します。
3. プロパティーを暗号化するには、**「暗号化」**ボックスを選択します。
4. 86 ページの『標準コネクター・プロパティーの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

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

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

コネクター・プロパティーの暗号化

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

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

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

更新メソッド

付録 A『*コネクターの標準構成プロパティー*』の 58 ページの『プロパティー値の設定と更新』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

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 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。

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

ご使用のブローカーが ICS の場合

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

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

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

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

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

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

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

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

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

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

ほとんどのコネクターの場合、選択可能な項目は「**最大限の努力**」のみです。

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

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

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

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

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

ご使用のブローカーが WAS の場合

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

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

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

関連付けられているマップ (ICS のみ)

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

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

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

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

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的**

  場合によっては、関連付けられたマップを明示的にバインドすることが必要になります。

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

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

  以下のをステップを実行して、マップを明示的にバインドします。

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

リソース (ICS)

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

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

メッセージング (ICS)

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

トレース/ログ・ファイル値の設定

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

ログとトレースの値を変更するには、以下のステップを実行します。

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

     注: STDOUT オプションは、Windows プラットフォームで実行しているコネクターの**「トレース/ログ・ファイル」**タブでのみ使用できます。
   - ファイルに:
     ログ・メッセージまたはトレース・メッセージを指定されたファイルに書き込みます。ファイルを指定するには、ディレクトリー・ボタン (省略符号) をクリックし、指定する格納場所に移動し、ファイル名を指定し、**「保管」**をクリックします。ログ・メッセージまたはトレース・メッセージは、指定した場所の指定したファイルに書き込まれます。

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

データ・ハンドラー

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

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

構成ファイルの保管

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

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

- System Manager から、統合コンポーネント・ライブラリーに *.con 拡張子付きファイルとして保管します。
- 指定したディレクトリーに保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに *.cfg 拡張子付きファイルとして保管します。デフォルトでは、このファイルは ¥WebSphereAdapters¥bin¥Data¥App に保管されます。
- WebSphere Application Server プロジェクトをセットアップしている場合には、このファイルを WebSphere Application Server プロジェクトに保管することもできます。

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

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

構成ファイルの変更

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

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

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

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

構成の完了

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

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

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

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

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

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

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

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

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

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

付録 C. Adapter for Telcordia のサンプル

サンプルについて

この付録で取り上げるサンプルは、Telcordia のオーダー応答を以下の形式で表現したものです。

- ビジネス・オブジェクト定義
- XML スキーマ
- XML

ビジネス・オブジェクト定義のサンプル

以下に、ビジネス・オブジェクト定義 Telcordia_OrderResponse を示します。

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

        [Attribute]
        Name = XMLDeclaration
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = false
        AppSpecificInfo = type=pi
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = ROOT
        Type = Telcordia_ROOT_OrderResponse
        ContainedObjectVersion = 1.0.0
        Relationship = containment
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = false
        AppSpecificInfo = OrderResponse
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = ObjectEventId
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = true
        IsForeignKey = false
        IsRequired = false
        IsRequiredServerBound = false
```

```
        [End]

        [Verb]
        Name = Create
        [End]
[End]
[BusinessObjectDefinition]
Name = Telcordia_ROOT_OrderResponse
Version = 1.0.0
AppSpecificInfo = OrderResponse

        [Attribute]
        Name = OrderResponse_Wrapper1
        Type = Telcordia_OrderResponse_OrderResponse_Wrapper1
        ContainedObjectVersion = 1.0.0
        Relationship = containment
        Cardinality = n
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = true
        AppSpecificInfo = (C0|CUR)
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = ObjectEventId
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = true
        IsForeignKey = false
        IsRequired = false
        IsRequiredServerBound = false
        [End]

        [Verb]
        Name = Create
        [End]
[End]
[BusinessObjectDefinition]
Name = Telcordia_OrderResponse_OrderResponse_Wrapper1
Version = 1.0.0

        [Attribute]
        Name = C0
        Type = Telcordia_OrderResponse_C0
        ContainedObjectVersion = 1.0.0
        Relationship = containment
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = true
        AppSpecificInfo = C0
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = CUR
        Type = Telcordia_OrderResponse_CUR
        ContainedObjectVersion = 1.0.0
        Relationship = containment
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = true
        AppSpecificInfo = CUR
```

```
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = ObjectEventId
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = true
        IsForeignKey = false
        IsRequired = false
        IsRequiredServerBound = false
        [End]

        [Verb]
        Name = Create
        [End]
[End]
[BusinessObjectDefinition]
Name = Telcordia_OrderResponse_C0
Version = 1.0.0
AppSpecificInfo = C0

        [Attribute]
        Name = C0_Wrapper1
        Type = Telcordia_OrderResponse_C0_Wrapper1
        ContainedObjectVersion = 1.0.0
        Relationship = containment
        Cardinality = n
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = true
        AppSpecificInfo = (CORS|DD|FT|ORDNO|OT|RSYS|TSYS|TT|WC)
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = ObjectEventId
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = true
        IsForeignKey = false
        IsRequired = false
        IsRequiredServerBound = false
        [End]

        [Verb]
        Name = Create
        [End]
[End]
[BusinessObjectDefinition]
Name = Telcordia_OrderResponse_C0_Wrapper1
Version = 1.0.0

        [Attribute]
        Name = CORS
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = false
        AppSpecificInfo = CORS;type=pcdata
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = DD
```

```
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = true
AppSpecificInfo = DD;type=pcdata
IsRequiredServerBound = false
[End]
[Attribute]
Name = FT
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = true
AppSpecificInfo = FT;type=pcdata
IsRequiredServerBound = false
[End]
[Attribute]
Name = ORDNO
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = true
AppSpecificInfo = ORDNO;type=pcdata
IsRequiredServerBound = false
[End]
[Attribute]
Name = OT
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = true
AppSpecificInfo = OT;type=pcdata
IsRequiredServerBound = false
[End]
[Attribute]
Name = RSYS
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = true
AppSpecificInfo = RSYS;type=pcdata
IsRequiredServerBound = false
[End]
[Attribute]
Name = TSYS
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = true
AppSpecificInfo = TSYS;type=pcdata
IsRequiredServerBound = false
[End]
[Attribute]
Name = TT
Type = String
```

```
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = true
        AppSpecificInfo = TT;type=pcdata
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = WC
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = true
        AppSpecificInfo = WC;type=pcdata
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = ObjectEventId
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = true
        IsForeignKey = false
        IsRequired = false
        IsRequiredServerBound = false
        [End]

        [Verb]
        Name = Create
        [End]
[End]
[BusinessObjectDefinition]
Name = Telcordia_OrderResponse_CUR
Version = 1.0.0
AppSpecificInfo = CUR

        [Attribute]
        Name = CUR_Wrapper1
        Type = Telcordia_OrderResponse_CUR_Wrapper1
        ContainedObjectVersion = 1.0.0
        Relationship = containment
        Cardinality = n
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = true
        AppSpecificInfo = (STAT|MSG)
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = ObjectEventId
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = true
        IsForeignKey = false
        IsRequired = false
        IsRequiredServerBound = false
        [End]

        [Verb]
        Name = Create
        [End]
[End]
```

```
[BusinessObjectDefinition]
Name = Telcordia_OrderResponse_CUR_Wrapper1
Version = 1.0.0

        [Attribute]
        Name = STAT
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = false
        AppSpecificInfo = STAT;type=pcdata
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = MSG
        Type = Telcordia_OrderResponse_MSG
        ContainedObjectVersion = 1.0.0
        Relationship = containment
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = false
        AppSpecificInfo = MSG
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = ObjectEventId
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = true
        IsForeignKey = false
        IsRequired = false
        IsRequiredServerBound = false
        [End]

        [Verb]
        Name = Create
        [End]
[End]
[BusinessObjectDefinition]
Name = Telcordia_OrderResponse_MSG
Version = 1.0.0
AppSpecificInfo = MSG

        [Attribute]
        Name = MSG_Wrapper1
        Type = Telcordia_OrderResponse_MSG_Wrapper1
        ContainedObjectVersion = 1.0.0
        Relationship = containment
        Cardinality = n
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = true
        AppSpecificInfo = (CODE|TEXT|ERRTAGPATH|LDEST|CATEGORY)
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = ObjectEventId
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = true
```

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

        [Verb]
        Name = Create
        [End]
[End]
[BusinessObjectDefinition]
Name = Telcordia_OrderResponse_MSG_Wrapper1
Version = 1.0.0

        [Attribute]
        Name = CODE
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = false
        AppSpecificInfo = CODE;type=pcdata
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = TEXT
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = false
        AppSpecificInfo = TEXT;type=pcdata
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = ERRTAGPATH
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = false
        AppSpecificInfo = ERRTAGPATH;type=pcdata
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = LDEST
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = false
        AppSpecificInfo = LDEST;type=pcdata
        IsRequiredServerBound = false
        [End]
        [Attribute]
        Name = CATEGORY
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = false
        IsForeignKey = false
        IsRequired = false
        AppSpecificInfo = CATEGORY;type=pcdata
        IsRequiredServerBound = false
```

```
        [End]
        [Attribute]
        Name = ObjectEventId
        Type = String
        Cardinality = 1
        MaxLength = 255
        IsKey = true
        IsForeignKey = false
        IsRequired = false
        IsRequiredServerBound = false
        [End]

        [Verb]
        Name = Create
        [End]
[End]
```

XML スキーマのサンプル

以下に、ビジネス・オブジェクト定義 Telcordia_OrderResponse に対応する XML スキーマを示します。

```
<?xml version = "1.0" encoding = "UTF-8"?>

<!-- Schema Identifier: BD-DELV-SPEC-040, Issue 2A, July 2002, Release 9.5.1-->
<!-- Last Modified: November 3, 2002-->

<xsd:schema xml:lang = "en" xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <xsd:include schemaLocation = "./CURTags.xsd"/>

   <xsd:element name = "C0">
      <xsd:annotation>
         <xsd:documentation>
FullName=[*C0 Route Control Header Section]
Desc=[This section contains routing data for a Customer Order. The *C0
section must be the first]
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:all>
            <xsd:element ref = "CORS" minOccurs = "0"/>
            <xsd:element ref = "DD"/>
            <xsd:element ref = "FT"/>
            <xsd:element ref = "ORDNO"/>
            <xsd:element ref = "OT"/>
            <xsd:element ref = "RSYS"/>
            <xsd:element ref = "TSYS"/>
            <xsd:element ref = "TT"/>
            <xsd:element ref = "WC"/>
         </xsd:all>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name = "CUR">
      <xsd:annotation>
         <xsd:documentation>
FullName=[Customer Order Response Section]
Desc=[This Customer Order Response section contains status information
and exception information to be returned to the order originator.]
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:all>
            <xsd:element ref = "STAT" minOccurs = "0"/>
            <xsd:element name = "MSG" minOccurs = "0">
```

```
               <xsd:annotation>
                  <xsd:documentation>
FullName=[Message Aggregate]
Desc=[The MSG aggregate contains exception information to be returned
to the order originator.]
Format=[ Conditional (required for error and non-involvement statuses;
not present if STAT is ACTV). This aggregate will appear once for each
exception message]
                  </xsd:documentation>
               </xsd:annotation>
               <xsd:complexType>
                  <xsd:all>
                     <xsd:element ref = "CODE" minOccurs = "0"/>
                     <xsd:element ref = "TEXT" minOccurs = "0"/>
                     <xsd:element ref = "ERRTAGPATH" minOccurs = "0"/>
                     <xsd:element ref = "LDEST" minOccurs = "0"/>
                     <xsd:element ref = "CATEGORY" minOccurs = "0"/>
                  </xsd:all>
               </xsd:complexType>
            </xsd:element>
         </xsd:all>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name = "OrderResponse">
      <xsd:complexType>
         <xsd:all>
            <xsd:element ref = "C0"/>
            <xsd:element ref = "CUR"/>
         </xsd:all>
      </xsd:complexType>
   </xsd:element>

</xsd:schema>
```

XML のサンプル

以下に、 102 ページの『XML スキーマのサンプル』に示した XML スキーマに対応する XML ファイルを示します。

```
<OrderResponse>

    <C0>

       <FT>COR</FT>

       <TT>SO</TT>

       <ORDNO>1COPY004</ORDNO>

       <OT>N</OT>

       <RSYS>SOP</RSYS>

       <TSYS>BND</TSYS>

       <WC>803921</WC>

       <DD>20020730</DD>

    </C0>

       <CUR>
          <STAT>EINP</STAT>
          <MSG>
              <CODE>0173</CODE>
```

```
                <TEXT>LSO NOT FOUND IN WC MAPPING REFERENCE TABLE</TEXT>
                <ERRTAGPATH>DELV.SR.RU.DATA.CKT.NEWACL.LSO</ERRTAGPATH>
            </MSG>
        </CUR>
</OrderResponse>
```

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。

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

商標

以下は、IBM Corporation の商標です。

IBM
IBM ロゴ
AIX
CrossWorlds
DB2
DB2 Universal Database
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、Pentium および ProShare は、Intel Corporation の米国およびその他の国における商標です。

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

WebSphere Business Integration Adapter Framework V2.4.0

Printed in Japan

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