IBM WebSphere Business Integration
Adapters

Adapter for QAD MFG/PRO ユーザーズ・ガイド

V 2.0.0

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、 91 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、Adapter for QAD MFG/PRO バージョン 2.0.0 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： IBM WebSphere Business Integration Adapters
Adapter for QAD MFG/PRO User Guide
V 2.0.0

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2刷 2004.3

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

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

© Copyright International Business Machines Corporation 2004. All rights reserved.

© Copyright IBM Japan 2004

目次

© Copyright IBM Corp. 2004

図

© Copyright IBM Corp. 2004

本書について

IBM® WebSphere® 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、およびレガシー・システムとメインフレーム・システムに、統合接続性を提供します。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、Adapter for QAD MFG/PRO のインストール、構成、およびビジネス・オブジェクト開発について説明します。

対象読者

本書は、お客様のサイトで WebSphere Business Integration システムを使用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

本書を利用するには、WebSphere Business Integration システム、ビジネス・オブジェクトとコラボレーションの開発、および QAD MFG/PRO アプリケーション・スイートについての知識が必要です。

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 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。

- アダプターの一般情報、WebSphere Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) におけるアダプターの使用、および WebSphere Application Server におけるアダプターの使用については、次のアドレスの IBM WebSphere Business Integration Adapters InfoCenter を参照してください。
  http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- WebSphere InterChange Server におけるアダプターの使用については、次のアドレスの IBM WebSphere InterChange Server InfoCenters 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
- WebSphere Message Brokers の詳細については、以下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

© Copyright IBM Corp. 2004

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。

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	変数名または相互参照を示します。
青のアウトライン	マニュアルをオンラインで表示するときにのみ見られる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。
< >	命名規則により、1 つの名前の個々の要素を互いに区別するために、不等号括弧によって個々の要素が囲まれます。例えば、`<server_name><connector_name>tmp.log` のように使用します。
/、¥	本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。UNIX システムの場合には、円記号をスラッシュ (/) に置き換えてください。すべての製品のパス名は、ご使用のシステムで製品がインストールされているディレクトリーを基準とした相対パス名です。
%text% および *$text*	% 記号で囲まれたテキストは、Windows の `text` システム変数またはユーザー変数の値を示します。UNIX 環境での同等の表記は *$text* です。これは、*text* UNIX 環境変数の値を示します。
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。

本リリースの新機能

リリース 2.0.x の新機能

2004 年 2 月

このアダプターは、MFG/PRO バージョン eB2 をサポートしています。

アダプターには、以下の変更点も盛り込まれています。

- **要求処理**
  - アダプターは、Web サービス・インターフェースを使用してブローカー要求を処理します。
  - 要求が同期化されました。
  - 要求のビジネス・オブジェクト構造が新しくなりました。
  - SQL Query Dispatcher は、新しい要求処理スタイルでは不要になったため、除去されました。
- **イベント処理**
  - アダプターは、トリプレットだけでなく、イベント処理時の QDoc-XML フォーマットもサポートします。

2003 年 12 月

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、第 2 章を参照してください。

バージョン 2.0.0 から、Adapter for QAD MFG/PRO は Microsoft Windows NT 上ではサポートされなくなりました。

リリース 1.0.x での新機能

本書は新規の資料です。

© Copyright IBM Corp. 2004

第 1 章 概要

Connector for QAD(TM) MFG/PRO(R) は、WebSphere Business Integration Adapter for QAD MFG/PRO のランタイム・コンポーネントの 1 つです。統合ブローカーは、コネクター・コンポーネントによって、WebSphere MQ メッセージの形式でデータを送受信する QAD MFG/PRO アプリケーションとビジネス・オブジェクトを交換できるようになります。

この章では、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。次のトピックがあります。

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。環境要件は、以下のセクションでリストされています。

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 QAD MFG/PRO のバージョン 2.0.0 は、以下のアダプター・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 2.2、2.3、2.4
- **統合ブローカー:**
  - WebSphere InterChange Server、バージョン 4.2、4.2.1、4.2.2
  - WebSphere MQ Integrator、バージョン 2.1.0
  - WebSphere MQ Integrator Broker、バージョン 2.1.0

© Copyright IBM Corp. 2004

- 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: 統合ブローカーおよびその前提条件のインストールに関する説明については、以下のガイドを参照してください。
WebSphere InterChange Server (ICS) については、「*IBM WebSphere InterChange Server システム・インストール・ガイド (UNIX 版)*」または「*IBM WebSphere InterChange Server システム・インストール・ガイド (Windows 版)*」を参照してください。
WebSphere Message Brokers 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、「*IBM WebSphere Business Integration Adapters アダプター実装ガイド (WebSphere Application Server)*」を参照してください。

アダプターのプラットフォームおよびデータベース

アダプターは以下のアプリケーション・プラットフォームおよびデータベースでサポートされています。

オペレーティング・システム:

- AIX 4.2.1、AIX 4.3、AIX 5.1、AIX 5.2
- Solaris 7.0、Solaris 8.0
- HP UX 11.0、HP UX 11i
- Windows 2000

データベース:

- Oracle RDBMS (8.1.7)
- Progress 9.1D+ Enterprise DB
- Progress 9.1D+ 4GL

サード・パーティーのソフトウェア:

- QAD MFG/PRO eB2

QXtend 製品および Q/LinQ MQSeries MOM 製品は QAD からインストールする必要がありますが、必ずしもアダプターと同じマシン上にインストールする必要はありません。

イベントを生成するには、QAD 製品の DataSync も必要です。この製品は Q/LinQ 上にインストールされますが、別個の製品として販売されています。

アダプターの依存関係

Adapter for QAD MFG/PRO は 3 ページの表 1 に示すように、クライアント・ライブラリーまたはAPI に依存しています。

表 1. アダプターの依存関係

ライブラリー/API	バージョン	オペレーティング・システム
IBM JRE	1.3.1.6	サポートされるすべての OS
MQSeries または JMS の Java ライブラリー	5.2+	サポートされるすべての OS
Xerces	1.4.3	サポートされるすべての OS
dom4j	1.4	サポートされるすべての OS
JNDI	1.2.1	サポートされるすべての OS
Apache SOAP	2.3.1	サポートされるすべての OS
WSDL4J	1.0	サポートされるすべての OS
IBM JSSE	1.0.2	サポートされるすべての OS

注: アダプターは、WebSphere MQ 5.3 の Secure Socket Layers (SSL) をサポートしていません。アダプター・フレームワーク統合ブローカー通信に適した WebSphere MQ ソフトウェア・バージョンについては、ご使用のプラットフォーム (Windows または Unix) の「インストール・ガイド」を参照してください。

ロケール依存データ

このアダプターは DBCS (2 バイト文字セット) で使用可能であり、翻訳されています。2 バイト文字セットをサポートし、指定の言語でメッセージ・テキストを配信できるように国際化されています。コネクターがある文字コード・セットを使用する場所から別の文字コード・セットを使用する場所にデータを転送する場合、文字変換を行ってデータの意味を保持します。

Java 仮想マシン (JVM) 内部の Java ランタイム環境では、Unicode 文字コード・セットでデータを表現します。Unicode は、既知の文字コード・セットのほとんど (単一バイトおよびマルチバイトの両方) に対応するエンコード方式を含んでいます。IBM WebSphere Business Integration システムのほとんどのコンポーネントは Java で書かれています。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、55 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

コネクター・アーキテクチャー

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合わせたコードが格納されています。コネクター・フレームワークのコードはすべてのコネクターに共通なので、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仲介役の機能を果たします。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントとの間で以下のようなサービスを提供します。

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

コネクターはメタデータ主導型です。コネクター構成プロパティー、およびビジネス・オブジェクト定義の役割を果たすメタオブジェクトで、メタデータ (ビジネス・オブジェクト属性とアプリケーション固有の処理に関する情報) を指定します。これにより、コネクター自体で指示をハードコーディングする必要がなくなります。コネクター用のメタデータの詳細については、17 ページの『コネクターの構成』および 31 ページの『アダプターのビジネス・オブジェクトの構造』を参照してください。

コネクターは WebSphere MQ メッセージングを使用してメッセージを受信し、SOAP/HTTP プロトコル・ベースのチャネルを使用してメッセージを送信します。詳細については、5 ページの『アプリケーションとコネクターの通信』を参照してください。以下の各セクションで、コネクターが情報をどのように処理するかを説明します。

コネクター処理

コネクターを使用すると、QAD MFG/PRO eB2 アプリケーションは、ビジネス・オブジェクトを以下との間で交換できます。

- IBM WebSphere Business Integration コラボレーション (WebSphere InterChange Server—ICS で構成した場合)
- メッセージ・フロー (WebSphere Integration Broker Message Brokers で構成した場合)
- Enterprise Java Beans (WebSphere Application Server で構成した場合)

図 1 には、上位概念でのコネクターの処理環境を示します。

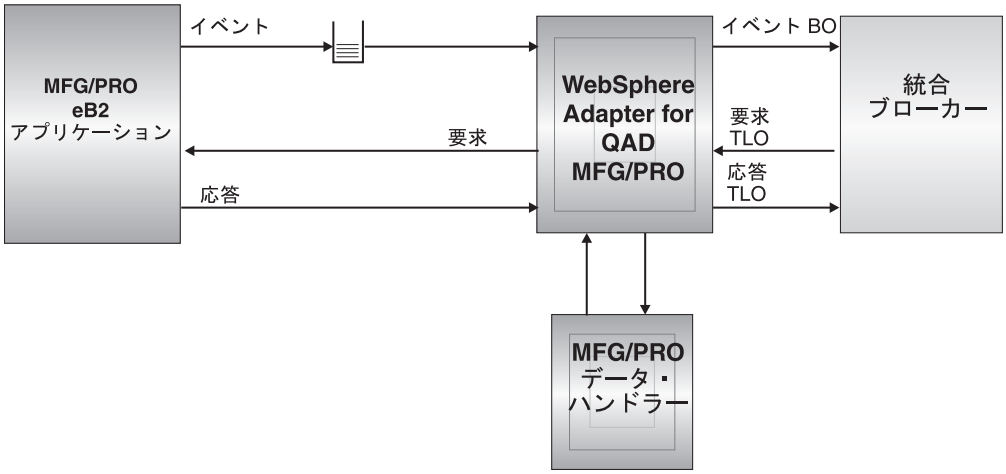

図 1. コネクター・アーキテクチャー

イベント処理の場合は、メッセージ・ルーティングとフォーマット変換がイベント・ポーリング技法で開始されます。イベントが検出されると、コネクターは QAD MFG/PRO キューからメッセージを検索します。コネクターから送信されるメッセージは、トリプレット・フォーマットや QDoc フォーマットで格納できます。フォーマットの詳細については、7 ページの『データ・フォーマット』を参照してください。コネクターは MFG/PRO データ・ハンドラーをインスタンス化して、メッセージを対応するビジネス・オブジェクト (BO) に変換します。その後コネクターは、BO を統合ブローカーに配信して、その後の処理に備えます。

要求処理は同期処理です。コネクターはトップレベル・オブジェクト (TLO) を統合ブローカーから受信します。特殊ビジネス・オブジェクトである TLO には、要求、応答、障害の各ビジネス・オブジェクトが格納されています。コネクターは、QAD MFG/PRO データ・ハンドラーを使用することにより、SOAP ヘッダーを追加して、要求 BO を SOAP/XML フォーマットに変換します。フォーマットの詳細については、7 ページの『データ・フォーマット』を参照してください。次に、コネクターは、SOAP/HTTP プロトコル・ベースのインターフェースを介して、メッセージを MFG/PRO アプリケーションに送信します。応答メッセージを受信すると、コネクターはこのメッセージを SOAP/XML からビジネス・オブジェクトに変換し、そのビジネス・オブジェクトを統合ブローカーに戻します。

イベントと要求の処理およびデータ・フォーマットの詳細については、『アプリケーションとコネクターの通信』を参照してください。

アプリケーションとコネクターの通信

イベント処理の場合、コネクターは、Java Message Service (JMS) のインプリメンテーションである IBM の WebSphere MQ メッセージング・レイヤーを使用します。JMS は、エンタープライズ・メッセージング・システムにアクセスするためのオープン・スタンダード API です。これは、ビジネス・アプリケーションがビジネス・データとイベントを非同期で送受信できるように設計されています。また、保証されたイベント・デリバリーを可能にします。コネクターは、MFG/PRO アプリケーションのイベント (出力) キューをポーリングしてイベントを検索します。

反対方向では、コネクターは QAD の QXtend SOAP/HTTP インターフェースを使用して要求を配置します。

QAD MFG/PRO アプリケーション・インターフェース

外部統合スイートの一部として (また、Adapter for QAD MFG/PRO とは別に)、QAD は、固有の MFG/PRO アプリケーションと Adapter for QAD 間のデータ転送に対応するインターフェースを備えています。このインターフェースには、コネクターのために 2 つのチャネルがあります。1 つはイベント処理用、もう 1 つは要求処理用です。

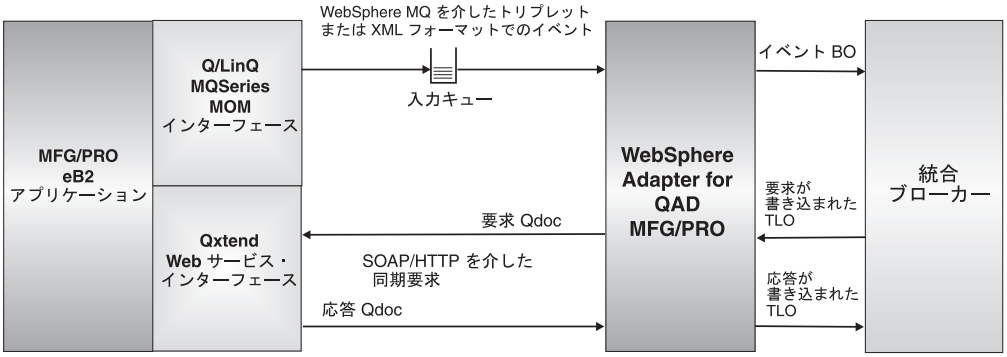

図 2. QAD MFG/PRO 外部インターフェース・アーキテクチャー

イベント処理時のインターフェース

図 2 に示すように、(QAD コンポーネント名が `QqMomAdapter` の) QAD インターフェースは、Q/LinQ MQSeries MOM (メッセージ指向ミドルウェア) インターフェースとして知られています。このインターフェースがインプリメントする MQSeries コンポーネントは、互換性のある前のバージョンの WebSphere MQ メッセージング・レイヤーに基づいています。

Q/LinQ MQSeries MOM インターフェース

- 固有の MFG/PRO アプリケーション・データと MQ メッセージとの変換を行います。
- Connector for QAD MFG/PRO が MFG/PRO アプリケーションからイベントを受信し、オプションで MFG/PRO アプリケーションに確認データまたはエラー・メッセージを送信するためのキュー (入力および出力) をサポートします。

入力キューは、MFG/PRO アプリケーションから WebSphere 接続アプリケーションへ渡されるイベントを処理するときに使用します。コネクターは ReplyTo キューを配置しません。

要求処理のインターフェース

要求処理の場合、コネクターは QAD の QXtend 統合フレームワークを使用します。QXtend フレームワークは、アプリケーション・サーバー上で実行される J2EE アプリケーションです。このアプリケーションは、SOAP-XML 1.2 フォーマットの要求を受け入れる Web サービス・インターフェースを備えています。

TLO がブローカーからコネクターに配信されると、コネクターは要求 BO を抽出して、これを SOAP エンベロープを持つ XML フォーマットに変換します。 次にコネクターは、SOAP-HTTP プロトコルを使用して同期要求を QXtend に発行します。応答を受信すると、コネクターはこれをビジネス・オブジェクトに再変換し、この応答 BO を TLO に書き込み、TLO をブローカーに送信します。

データ・フォーマット

コネクターが処理するデータ・フォーマットは、MFG/PRO アプリケーションがサポートするネイティブ・フォーマットに対応しています。図 3 にイベント処理のデータ・フォーマットを示します。

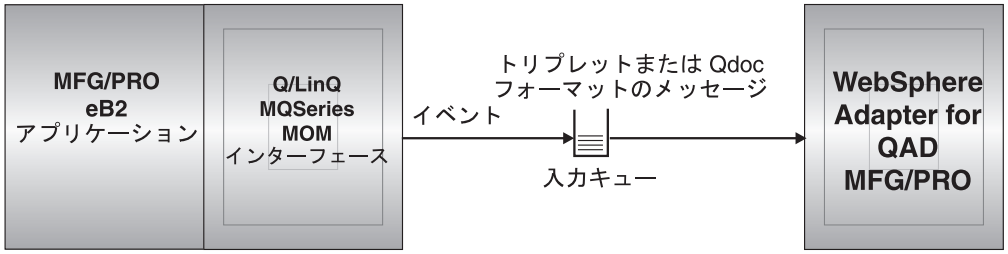

図 3. QAD MFG/PRO データ・フォーマット—イベント処理

イベント処理フォーマット QAD MFG/PRO eB2 アプリケーションからのイベント・メッセージは、トリプレット・フォーマットまたは QDoc フォーマット (いずれも QAD 専有) で公開されます。トリプレット・フォーマットは、タグ付きフィールド値がシリアライズされた単一の ASCII テキスト・ストリングからなります。QDoc フォーマットは、XML ベースのフォーマットです。(トリプレット・フォーマットや QDoc フォーマットの詳細については、QAD の資料を参照してください。) コネクターは、いずれかのフォーマットで作成されたメッセージをビジネス・オブジェクトに変換し、ブローカーに配信します。

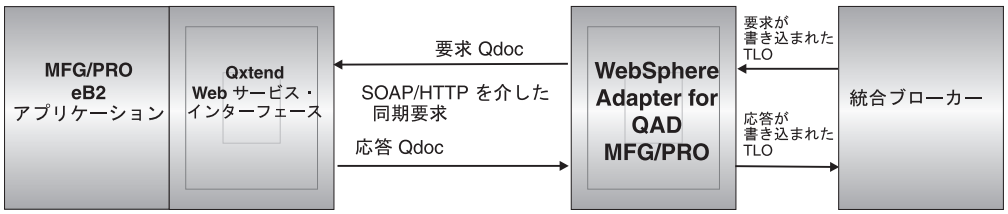

図 4. QAD MFG/PRO データ・フォーマット—要求処理

要求処理フォーマット コネクターは、ブローカーから TLO を受信します。TLO には、要求、応答、障害の各ビジネス・オブジェクトが子として格納されています。コネクターは要求オブジェクトを QDoc メッセージに変換します。QDoc は、専有 XML フォーマットです。QDoc のペイロードは、SOAP 1.2 構文に準拠した QAD 固有のメッセージ・エンベロープでラップされます。エンベロープのルート要素には、ヘッダーと本文コンポーネントが含まれています。ヘッダーの構造は、すべての QDoc 文書に共通です。QDoc 要求メッセージは、SOAP/HTTP プロトコルを介して QXtend Web サービス・モジュールに送信されます。MFG/PRO eB2 アプリケーションが応答メッセージを戻すと、コネクターはそのメッセージを応答ビジ

ネス・オブジェクトに変換します。この後、TLO はブローカーに戻ります。QDoc フォーマットの詳細については、QAD の QDoc 仕様を参照してください。TLO およびビジネス・オブジェクト構造の詳細については、31 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。

コネクターのデータ・ハンドラーによるメッセージおよびビジネス・オブジェクトの処理方法の詳細については、41 ページの『第 4 章 QAD MFG/PRO データ・ハンドラー』を参照してください。

メッセージ要求

図 5 に、メッセージ要求の通信を示します。

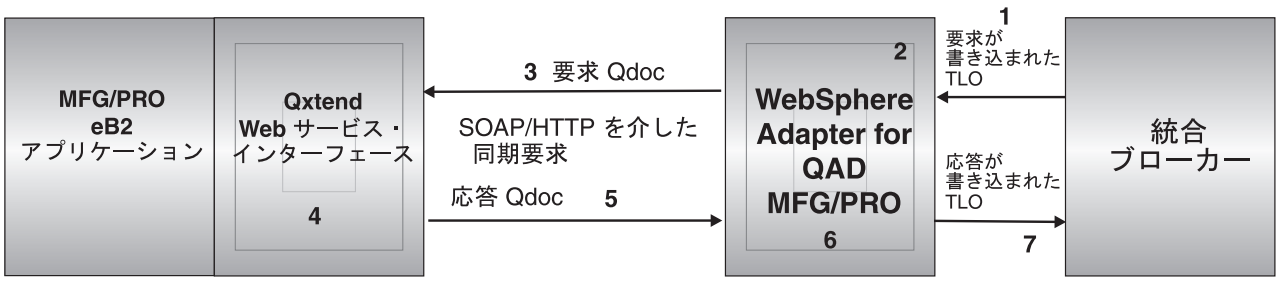

図 5. アプリケーションとコネクターの通信方式: 要求処理

1. 統合ブローカーは、要求ビジネス・オブジェクトが格納されている TLO をコネクターに送信します。
2. コネクターのデータ・ハンドラーは、要求 BO を、QDoc フォーマットの SOAP-XML メッセージに変換します。
3. コネクターは、SOAP/HTTP を介して QDoc 要求メッセージを QXtend Web サービス・インターフェースに送信します。
4. QXtend は、この要求を QAD MFG/PRO eB2 アプリケーションに送信して、応答を待ちます。QXtend は応答 SOAP-XML メッセージを QDoc フォーマットで生成します。
5. QXtend は応答メッセージをコネクターに送信します。
6. コネクターのデータ・ハンドラーは、メッセージを応答ビジネス・オブジェクトに変換し、最初ブローカーから受信した TLO の中にこのメッセージを置きます。
7. コネクターは TLO をブローカーに戻します。

イベント・デリバリー

図 6 に、イベント・デリバリーの方向を示します。

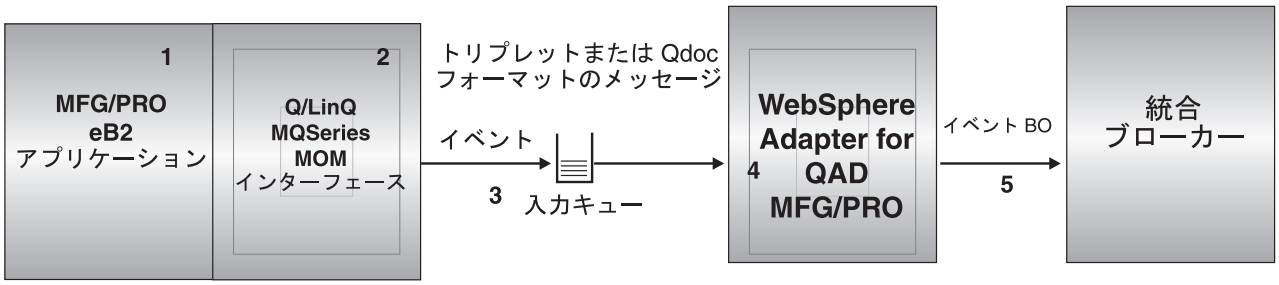

図 6. アプリケーションとコネクターの通信方式: イベント・デリバリー

1. QAD MFG/PRO eB2 のデータが変更されます。
2. Q/LinQ MQSeries MOM インターフェースは、イベント・メッセージを生成して、これを MQ キューに置きます。
3. コネクターの `pollForEvents()` メソッドは、入力キューで次のメッセージを検索します。
4. 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。メッセージが QAD MFG/PRO データ・ハンドラーに渡されます。データ・ハンドラーは、メッセージがトリプレット・フォーマットと QDoc フォーマットのどちらで作成されているかを判別します。メッセージがトリプレット・フォーマットで作成されている場合、データ・ハンドラーはトリプレットから XML へのマッピング・エンジンを起動し、マッピング・エンジンはトリプレット・メッセージを QDoc 文書にマップします。次にデータ・ハンドラーは QDoc 文書を WebSphere のビジネス・オブジェクトに変換します。そうではなく、メッセージが QDoc フォーマットで作成されている場合、データ・ハンドラーはトリプレットから XML へのマッピング・ステップをスキップし、QDoc をビジネス・オブジェクトに変換します。
5. コネクターの `gotApplEvents()` メソッドは、このビジネス・オブジェクトを統合ブローカーに配信して次の処理を実行します。さらに、メッセージは進行中のキューから除去されます。

イベント処理

コネクターはアプリケーションの出力キューをポーリングして、データベース・トリガーを介して入力キューに書き込まれたイベントを検出します。データベース・トリガーは、コネクター・イベント処理用に構成する MFG/PRO アプリケーションごとにインストールしてください。トリガーは、QAD が提供する DataSync モジュールを使用してインストールします。詳細については、QAD の資料 (「*External Interface Guide — Data Synchronization*」) を参照してください。イベントは、MFG/PRO アプリケーションが MQ メッセージを生成し、それを入力キューに格納したときに発生します。

検索

コネクターは `pollForEvents()` メソッドを使用して、メッセージに対し、通常の間隔で入力キューをポーリングします。コネクターは、メッセージを検出すると、それを入力キューから取り出します。コネクターは、構成されたデータ・ハンドラー

にメッセージを渡し、データ・ハンドラーは、ビジネス・オブジェクトを作成し値を入力して動詞を指定します。イベントの失敗のシナリオについては、37 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。トランザクションを使用するこの方法では、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がキュー内でのトランザクションのコミットに失敗した場合に、ビジネス・オブジェクトが 1 つのコラボレーションに 2 回送信される可能性があります。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ずに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。そのため、コネクターはキューからメッセージを取り出すと、1) メッセージがビジネス・オブジェクトに変換されるか、2) `gotApplEvents()` メソッドによってビジネス・オブジェクトが統合ブローカーに送信されるか、3) 戻り値が受信されるまで、検索にコミットしません。

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 (始動時の失敗、再処理、無視、またはエラー・ログの記録) のうちの 1 つを指定できます。

始動時の失敗

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。

再処理

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。

無視

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。

エラー・ログ

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。

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・フレームワークでは、イベントの消失を防止することができます。

注: 保証付きイベント・デリバリー機能を使用しない場合、コネクターがイベントをパブリッシュする時間 (コネクターが gotApplEvent() メソッドを自身の pollForEvents() メソッド内部で呼び出す時間) と、コネクターがイベント・レコードを削除することによってイベント・ストアを更新する (または「イベント送付済み」状況を使用して更新する) 時間との間のわずかな期間に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。コネクターは再始動時にイベント・ストア内に残ったイベント・レコードを検出して送信するため、結果的にイベントが 2 回送信されることになります。

保証付きイベント・デリバリーを使用するためにコネクターを構成するには、25 ページの『保証付きイベント・デリバリーの使用可能化』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・オブジェクトの作成

ビジネス・オブジェクトを作成または変更するには、Object Discovery Agent (ODA) を使用します。XML ODA は、ビジネス・オブジェクト作成のプロセスを自動化します。詳細については、31 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。

ビジネス・オブジェクト要求

ビジネス・オブジェクト要求は、統合ブローカーがビジネス・オブジェクトを `doVerbFor()` メソッドに送信したときに処理されます。コネクターは、構成済みのデータ・ハンドラーを使用して、ビジネス・オブジェクトを SOAP/XML フォーマットの QAD MFG/PRO QDoc メッセージに変換します。

動詞の処理

コネクターは、各ビジネス・オブジェクトの動詞に基づいてビジネス・オブジェクトを処理します。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。コネクターは以下のビジネス・オブジェクトの動詞をサポートします。

- Create
- Update
- Delete

一般的なインストール作業および構成作業

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作業と始動作業について概説します。

統合ブローカーのインストール

詳細については、1 ページの『ブローカーの互換性』を参照し、さらに使用している統合ブローカーについてはインストール・ガイドを参照してください。

WebSphere Business Integration Adapter Framework のインストール

詳細については、1 ページの『ブローカーの互換性』を参照してください。

WebSphere Business Integration Data Handler for XML のインストール

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。

QAD 製品のインストール

QXtend 製品および Q/LinQ MQSeries MOM 製品は QAD からインストールする必要があります。ただし、これらの製品をアダプターと同じマシンにインストールする必要はありません。

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、15 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。

メタデータの定義

以下についてメタデータを定義する必要があります。

- トリプレットから XML へのマッピング・ファイル。ファイルのサンプルは提供されています。サンプルの詳細については、16 ページの『インストール済みファイルの構造』を参照してください。トリプレットから XML へのマッピングについては、41 ページの『第 4 章 QAD MFG/PRO データ・ハンドラー』を参照してください。
- 以下のメタオブジェクト。
  - アウトバウンド・マッピング・メタオブジェクト。詳細については、42 ページの『QAD MFG/PRO データ・ハンドラーの構成』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。詳細については、42 ページの『QAD MFG/PRO データ・ハンドラーの構成』を参照してください。

コネクターの構成

以下のセクションでは、インストール済みの Connector for QAD MFG/PRO を構成するために実行する必要がある作業について説明します。

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。

Connector Configurator を使用して、コネクター・プロパティーを設定することができます。

- Connector Configurator の詳細および操作手順については、73 ページの『付録 B. Connector Configurator』を参照してください。
- 標準のコネクター・プロパティーについては、55 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。
- コネクター固有のプロパティーについては、15 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。

次の統合ブローカーのいずれかと連動するようにコネクターを構成する必要があります。

- InterChange Server 統合ブローカー。概要については、「*IBM WebSphere InterChange Server テクニカル入門*」を参照してください。アダプターと ICS が連動するように構成する方法については、「*WebSphere InterChange Server インプリメンテーション・ガイド*」を参照してください。
- WebSphere MQ Integrator Broker。詳細については、「*WebSphere MQ Integrator Broker 用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 (WAS)。詳細については、「*アダプター実装ガイド (WebSphere Application Server)*」を参照してください。

Adapter for QAD MFG/PRO に対してコネクター・プロパティーを構成する場合は、以下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、WebSphere MQ サーバーのホストの名前と一致している。
- 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の値と一致すること。
- 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・プロパティーのキュー URI である InputQueue、InProgressQueue、および ErrorQueue が有効で実際に存在しており、InputQueue の値が、Q/LinQ MQSeries MOM (メッセージ指向ミドルウェア) インターフェースによってイベントが配信される MQ キューに設定されている。

Qxtend Web サービスの URL を構成するには、次の手順に従います。

- Business Object Designer の BIA_Protocol_ConfigMO を開き、その Destination 属性を、Qxtend Web サービスが実行されている場所の URL に設定します。詳細については、36 ページの『要求処理の TLO の生成: ステップバイステップ』を参照してください。

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 DataHandlerMetaObjectName で、構成を定義するデータ・ハンドラー・メタオブジェクトを指定します。詳細については、18 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。

データ・ハンドラーの処理の詳細については、41 ページの『第 4 章 QAD MFG/PRO データ・ハンドラー』を参照してください。

データベース・トリガーのインストール

コネクター・イベント処理用に構成する MFG/PRO アプリケーションごとにデータベース・トリガーをインストールします。トリガーは、QAD が提供する DataSync モジュールを使用してインストールします。詳細については、QAD の資料 (「*External Interface Guide — Data Synchronization*」) を参照してください。

QDoc スキーマのダウンロード

ビジネス・オブジェクトを作成するには、QDoc スキーマを使用します。QAD MFG/PRO から QDoc スキーマをダウンロードする手順などについては、34 ページの『イベント処理のビジネス・オブジェクト定義の生成: ステップバイステップ』または 36 ページの『要求処理の TLO の生成: ステップバイステップ』を参照してください。先に QDoc スキーマをダウンロードしてインストールしないかぎり、コネクターの始動や実行はできません。詳細については、15 ページの『アダプターと関連ファイルのインストール』を参照してください。

ビジネス・オブジェクト定義の作成

XML Object Discovery Agent (ODA) を使用してビジネス・オブジェクトを作成します。6 段階の自動化プロセスについて、31 ページの『第 3 章 ビジネス・オブジェクトの作成と変更』で説明します。ビジネス・オブジェクトを作成したら、コネクターによってサポートされているビジネス・オブジェクトのリストに、これらの定義をメタオブジェクトとともに追加します。詳細については、84 ページの『サポートされるビジネス・オブジェクト定義の指定』を参照してください。

第 2 章 コネクターのインストールと構成

この章では、コネクターをインストールおよび構成する方法と、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について説明します。本章の内容は、次のとおりです。

ブローカーの互換性、プラットフォームのサポート、依存関係などのアダプター環境の詳細については、1 ページの『アダプター環境』を参照してください。

インストール作業の概要

QAD MFG/PRO 用のコネクターをインストールするには、以下の作業を実行する必要があります。

- **統合ブローカーのインストール** WebSphere Business Integration システムのインストールと統合ブローカーの始動が含まれます。この作業は、ご使用のブローカーおよびオペレーティング・システムのインストール資料で説明されています。
- **XML データ・ハンドラーのインストール** 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。
- **アダプター・フレームワークのインストール** 詳細については、「*WebSphere Business Integration Adapters インストール・ガイド*」を参照してください。
- **アダプターおよび関連ファイルのインストール** この作業では、アダプターのファイルをソフトウェア・パッケージから使用システムにインストールします。『アダプターと関連ファイルのインストール』を参照してください。

アダプターと関連ファイルのインストール

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

注: コネクターを始動して実行するだけでなく、ビジネス・オブジェクトを生成するには、必要な QDoc スキーマを QAD Web サイトからダウンロードするか、または QXtend 製品から QDoc スキーマ・ファイルを取得する必要があります。その後、すべての .xsd ファイルを *<QXtend installation directory>*/webapps/qxtendserver/WEB-INF/schemas/eB2 から

© Copyright IBM Corp. 2004

ProductDir/dependencies/QDocSchemas にコピーします。ステップバイステップのダウンロード手順については、 34 ページの『イベント処理のビジネス・オブジェクト定義の生成: ステップバイステップ』または 36 ページの『要求処理の TLO の生成: ステップバイステップ』を参照してください。

インストール済みファイルの構造

以降のセクションでは、Windows および UNIX のインストール済みファイルの構造について説明します。

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¥QAD 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¥QAD¥BIA_QAD.jar	QAD MFG/PRO コネクターが使用するクラスを含みます。
connectors¥QAD¥start_mfgpro.bat	コネクターの始動スクリプト (NT/2000)。
connectors¥messages¥BIA_QADConnector.txt	コネクターのメッセージ・ファイル。
repository¥QAD¥BIA_CN_QAD_TEMPLATE	コネクターのリポジトリー定義。
connectors¥QAD¥dependencies¥BIA_HeaderMetaData.txt	QDoc ヘッダーを CIM/トリプレット・メッセージ・ヘッダーにマッピングするためのメタデータ。
connectors¥QAD¥dependencies¥BIA_qdocTemplate.xml	QDoc XML アウトバウンド・メッセージを作成する際に使用するテンプレート。
connectors¥QAD¥dependencies¥BIA_soapEnvelopeTemplate.xml	QDoc XML インバウンド・メッセージを作成する際に使用するテンプレート。
connectors¥QAD¥dependencies¥BIA_TripletMappingMDOutbound.xml	トリプレットと XML の変換用のメタデータ。
connectors¥QAD¥dependencies¥dom4j.jar	dom4j ライブラリー・クラス
connectors¥QAD¥dependencies¥mail.jar	Web サービスの対話のためのクラス
connectors¥QAD¥samples	サンプルのビジネス・オブジェクト定義およびテンプレート

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

UNIX のコネクター・ファイル構造

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

このユーティリティーにより、コネクターが *ProductDir*/connectors/QAD 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/QAD/BIA_QAD.jar	QAD MFG/PRO コネクターが使用するクラスを含みます。
connectors/QAD/start_QAD.sh	コネクターの始動スクリプト (AIX、Solaris、HP UX)。
connectors/messages/BIA_QADConnector.txt	コネクターのメッセージ・ファイル。
repository/QAD/BIA_CN_QAD_TEMPLATE	コネクターのリポジトリー定義。
connectors/QAD/dependencies/BIA_HeaderMetaData.txt	QDoc ヘッダーをトリプレット・メッセージ・ヘッダーにマッピングするためのメタデータ。
connectors/QAD/dependencies/BIA_qdocTemplate.xml	QDoc XML アウトバウンド・メッセージを作成する際に使用するテンプレート。
connectors/QAD/dependencies/BIA_soapEnvelopeTemplate.xml	QDoc XML インバウンド・メッセージを作成する際に使用するテンプレート。
connectors/QAD/dependencies/BIA_TripletMappingMDOutbound.xml	トリプレットと XML の変換用のメタデータ。
connectors/QAD/dependencies/dom4j.jar	dom4j ライブラリー・クラス
connectors/QAD/dependencies/mail.jar	Web サービスの対話のためのクラス
connectors/QAD/samples	サンプルのビジネス・オブジェクト定義およびテンプレート

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

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・プロパティーを設定することができます。

- Connector Configurator の詳細および操作手順については、73 ページの『付録 B. Connector Configurator』を参照してください。
- 標準のコネクター・プロパティーについては、18 ページの『標準コネクター・プロパティー』および 55 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。
- コネクター固有のプロパティーについては、18 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。

アダプターは、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ことができます。AgentTraceLevel のように、変更がすぐに有効になるコネクター構成プロパティーもあります。変更後にコンポーネントの再始動やシステム再始動が必要なコネクター・プロパティーもあります。プロパティーが動的 (すぐに有効になる) か静的 (コネクター・コンポーネントの再始動やシステム再始動が必要) かを判別する方法については、System Manager の「コネクター・プロパティー (Connector Properties)」ウィンドウの「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、すべてのアダプターによって使用される情報が提供されます。これらのプロパティーの資料については、 55 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。

コネクターを実行する前に、ApplicationName 構成プロパティーに値を設定する必要があります。

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。また、コネクター固有のプロパティーを使用すれば、エージェントを再コーディングまたは再ビルドせずに、コネクター内の静的情報やロジックを変更できます。

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	*WebSphere MQ のログイン・パスワード*		なし
ApplicationUserName	*WebSphere MQ のログイン・ユーザー ID*		なし
CCSID	*キュー・マネージャー接続用文字セット*	`null`	なし
Channel	*コネクターが WebSphere MQ と通信するために経由する MQ サーバー・チャネル*		はい
DataEncoding	*MO キューから Java ストリングへメッセージのバイトを変換するためのエンコード方式:* *US-ASCII*、*ISO-8859-1*、*UTF-8*、*UTF-16*	UTF-8	なし
DataHandlerMetaObjectName	*データ・ハンドラー・メタオブジェクト*	`BIA_DataHandlerConfigMO`	はい
ErrorQueue	*未処理のメッセージのキュー*		なし
HostName	*WebSphere MQ サーバー*		はい
InDoubtEvents	`FailOnStartup`、`Reprocess`、`Ignore`、`LogError`	`Reprocess`	なし
InputQueue	*MFG/PRO がイベントを書き込むキュー*		はい
InProgressQueue	*進行中イベント・キュー*		なし
PollQuantity	*InputQueue プロパティーで指定された各キューから検索するメッセージの数*	1	なし
Port	*WebSphere MQ リスナーのために確立するポート*		はい
ProtocolHandlerFramework	*これは階層プロパティーであり、値はありません。*	なし	はい
+ProtocolHandlers	*これは階層プロパティーであり、値はありません。*		はい

名前	指定可能な値	デフォルト値	必須
++SOAPHTTPHTTPSHandler	*これは階層プロパティーです。このプロパティーのサブプロパティーについては、 21 ページの『SOAPHTTPHTTPSHandler』を参照してください。*		はい
ProxyServer	*これは階層プロパティーであり、値はありません。*		なし
+HttpProxyHost	*HTTP プロキシー・サーバーのホスト名*		なし
+HttpProxyPort	*HTTP プロキシー・サーバーのポート番号*	80	なし
+HttpNonProxyHosts	*直接接続が必要な HTTP ホスト*		なし
+HttpsProxyHost	*HTTPS プロキシー・サーバーのホスト名*		なし
+HttpsProxyPort	*HTTPS プロキシー・サーバーのポート番号*	443	なし
+HttpsNonProxyHosts	*直接接続が必要な HTTPS ホスト*		なし
+SocksProxyHost	*Socks プロキシー・サーバーの名前*		なし
+SocksProxyPort	*Socks プロキシー・サーバーのポート*		なし
+HttpProxyUsername	*HTTP プロキシー・サーバー・ユーザー名*		なし
+HttpProxyPassword	*HTTP プロキシー・サーバー・パスワード*		なし
+HttpsProxyUsername	*HTTPS プロキシー・サーバー・ユーザー名*		なし
+HttpsProxyPassword	*HTTPS プロキシー・サーバー・パスワード*		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*		なし
UseDefaults	true または false	false	

ApplicationPassword

WebSphere MQ へのログイン時に UserID とともに使用されるパスワードです。

デフォルト = なし。

ApplicationPassword がブランクの場合または除去された場合、コネクターは、WebSphere MQ* が提供するデフォルトのパスワードを使用します。

ApplicationUserName

WebSphere MQ へのログイン時に Password とともに使用されるユーザー ID。

デフォルト = なし。

ApplicationUserName がブランクの場合または除去された場合、コネクターは、WebSphere MQ* が提供するデフォルトのユーザー ID を使用します。

CCSID
キュー・マネージャー接続の文字セット。プロパティーの値は、キュー URI 内の CCSID プロパティーの値と一致している必要があります。

デフォルト = null。

Channel
コネクターが WebSphere MQ と通信するために経由する MQ サーバー・コネクター・チャネルです。

デフォルト = なし。

Channel がブランクの場合または除去された場合、コネクターは、WebSphere MQ が提供するデフォルトのサーバー・チャネルを使用します。

DataEncoding
MO キューから Java ストリングへメッセージのバイトを変換するためのエンコード方式。使用可能な値は、US-ASCII、ISO-8859-1、UTF-8、UTF-16 です。

デフォルト = UTF-8

DataHandlerMetaObjectName
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トップレベル・メタオブジェクトの名前。詳細については、42 ページの『QAD MFG/PRO データ・ハンドラーの構成』を参照してください。

デフォルト = BIA_DataHandlerConfigMO

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 = なし。

HostName
WebSphere MQ のホストであ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

MFG/PRO がイベントを書き込むキューです。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。InputQueue は 1 つしかありません。

デフォルト = なし。

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= なし。

PollQuantity

pollForEvents スキャン中に InputQueue プロパティーで指定された各キューから検索するメッセージの数。

デフォルト = 1

Port

WebSphere MQ リスナーのために確立するポート。

デフォルト = なし。

ProtocolHandlerFramework

プロトコル・ハンドラー・フレームワークは、そのプロトコル・ハンドラーをロードおよび構成するために、このプロパティーを使用します。これは階層プロパティーであり、値はありません。

デフォルト = なし。

ProtocolHandlers

この階層プロパティーには、値がありません。この第 1 レベルの子は、別個のプロトコル・ハンドラーを表します。

デフォルト = なし。

SOAPHTTPHTTPSHandler

SOAP/HTTP-HTTPS プロトコル・ハンドラーの名前。これは階層プロパティーです。リスナーとは異なり、プロトコル・ハンドラーは重複しない場合もあるので、それぞれのプロトコルごとにハンドラーは 1 つだけしかないことがあります。表 2 は SOAP/HTTP-HTTPS プロトコル・ハンドラーのサブプロパティーを表しています。+ 文字は、プロパティー階層内の項目の位置を示しています。

表 2. SOAP/HTTP-HTTPS プロトコル・ハンドラー構成プロパティー

名前	指定可能な値	デフォルト値	必須
++SOAPHTTPHTTPSHandler	これは階層プロパティーであり、値はありません。		はい
+++Protocol	ハンドラーがインプリメントしようとしているプロトコルの種類。*SOAP/HTTP* および *SOAP/HTTPS* の場合の値は `soap/http` です。 **注:** このプロパティーの値を指定しなければ、コネクターはプロトコル・ハンドラーを初期化しません。		はい
+++HTTPReadTimeout	*リモート・ホスト (Web サービス) から読み取るときのタイムアウト・インターバル (ミリ秒単位) を指定する SOAP/HTTP 固有のプロパティー。このプロパティーを指定しなかった場合または 0 に設定した場合、SOAP/HTTP プロトコル・ハンドラーは、リモート・ホストからの読み取り中、無期限に妨害します。*	`0`	なし

ProxyServer

ネットワーク上で、アダプターが動作しているマシンと Tomcat QXtend サーバーとの間にプロキシー・サーバーを使用している場合は、このプロパティーの下の値を構成します。このプロパティーは階層型のプロパティーで、値はありません。このプロパティーの下で指定された値は、SOAP/HTTP/HTTPS プロトコル・ハンドラーによって使用されます。

HttpProxyHost

HTTP プロキシー・サーバーのホスト名。ネットワークが HTTP プロトコルでプロキシー・サーバーを使用する場合には、このプロパティーを指定してください。

デフォルト = なし。

HttpProxyPort

HTTP プロキシー・サーバーに接続するためにコネクターが使用するポート番号。

デフォルト = 80。

HttpNonProxyHosts

このプロパティーの値は、プロキシー・サーバー経由ではなく直接接続する必要のある、1 つまたは複数のホスト (HTTP の場合) を表します。この値は、各ホストを ″|″ で区切った、ホストのリストの形で指定することができます。

デフォルト = なし。

HttpsProxyHost

HTTPS プロキシー・サーバーの場合のホスト名。

デフォルト = なし。

HttpsProxyPort
HTTPS プロキシー・サーバーに接続するためにコネクターが使用するポート番号。

デフォルト = 443。

HttpsNonProxyHosts
このプロパティーの値は、プロキシー・サーバー経由ではなく直接接続する必要のある、1 つまたは複数のホスト (HTTPS の場合) を表します。この値は、各ホストを ″|″ で区切った、ホストのリストの形で指定することができます。

デフォルト = なし。

SocksProxyHost
Socks Proxy サーバーの場合のホスト名。ネットワークが Socks プロキシーを使用する場合には、このプロパティーを指定してください。

注: 基礎となる JDK は Socks をサポートしていなければなりません。

デフォルト = なし。

SocksProxyPort
Socks Proxy サーバーに接続するためのポート番号。ネットワークが Socks プロキシーを使用する場合には、このプロパティーを指定してください。

デフォルト = なし。

HttpProxyUsername
HTTP プロキシー・サーバーのユーザー名。Web サービス要求の宛先が HTTP URLの場合、「ProxyServer」->「HttpProxyUsername」を指定すると、プロキシーによる認証時に、SOAP HTTP/HTTPS プロトコル・ハンドラーによって Proxy-Authorization ヘッダーが作成されます。ハンドラーは、認証に CONNECT メソッドを使用します。

proxy-authentication ヘッダーは Base64 でエンコードされており、次の構造を持っています。
Proxy-Authorization: Basic
Base64EncodedString

ハンドラーは、ユーザー名とパスワード・プロパティー値 (コロン (:) によって区切られた) を連結し、Base64 でエンコードされたストリングを作成します。

デフォルト = なし。

HttpProxyPassword
HTTP プロキシー・サーバーのパスワード。これらの値の使用に関する詳細については、『HttpProxyUsername』を参照してください。

デフォルト = なし。

HttpsProxyUsername

HTTPS プロキシー・サーバーのユーザー名。Web サービス要求の宛先が HTTPS URLの場合、「ProxyServer」->「HttpsProxyUsername」を指定すると、プロキシーによる認証のために、SOAP HTTP/HTTPS プロトコル・ハンドラーによって Proxy-Authorization ヘッダーが作成されます。ハンドラーは、HttpsProxyUsername と HttpsProxyPassword の構成プロパティー値 (コロン (:) によって区切られた) を連結し、Base64 でエンコードされたストリングを作成します。

デフォルト = なし。

HttpsProxyPassword

HTTPS プロキシー・サーバーのパスワード。これらの値の使用に関する詳細については、『HttpsProxyUsername』を参照してください。

デフォルト = なし。

UnsubscribedQueue

サブスクライブされていないメッセージが送信されるキューです。

デフォルト = なし。

注: *WebSphere MQ が提供する値が誤っていたり、不明である可能性があるので、必ずこれらの値をチェックしてください。値が誤っていたり不明の場合は、値を暗黙的に指定してください。

UseDefaults

要求ビジネス・オブジェクトの場合、UseDefaults を `true` に設定すると、isRequired ビジネス・オブジェクト属性ごとに、有効な値とデフォルト値のどちらが指定されているかがコネクターによって確認されます。値が指定されている場合は、要求操作が続行されます。パラメーターが `false` に設定されている場合、コネクターによる確認は有効な値に対してのみとなり、値が指定されていない場合の要求操作は失敗します。デフォルト値は `false` です。

イベント処理の構成

Adapter for QAD MFG/PRO に対してコネクター・プロパティーを構成する場合は、以下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、WebSphere MQ サーバーのホストの名前と一致している。
- 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の値と一致すること。
- 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・プロパティーのキュー URI である InputQueue、InProgressQueue、および ErrorQueue が有効で実際に存在しており、InputQueue の値が、Q/LinQ MQSeries MOM (メッセージ指向ミドルウェア) インターフェースによってイベントが配信される MQ キューに設定されている。

要求処理の構成

要求処理の場合は、QXtend Web サービスの URL を指定する必要があります。これを実行するには、この URL を、要求 BO 定義の ProtocolConfigMO に存在する Destination 属性の値として指定します。

ProtocolConfigMOs の Destination 属性の設定の詳細については、34 ページの『要求処理の TLO 生成の概要』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 DataHandlerMetaObjectName で、構成を定義するデータ・ハンドラー・メタオブジェクトを指定します。詳細については、20 ページの『DataHandlerMetaObjectName』を参照してください。

データ・ハンドラーの処理の詳細については、41 ページの『第 4 章 QAD MFG/PRO データ・ハンドラー』を参照してください。

保証付きイベント・デリバリーの使用可能化

Adapter for QAD 用の保証付きイベント・デリバリー機能を構成することができます。そのためには、重複イベントの発生を防ぐ重複イベント回避機能を使用する必要があります。このセクションでは、保証付きイベント・デリバリー機能の使用に関して、以下のトピックを取り上げます。

- 『コネクター用の機能の使用可能化』
- 26 ページの『イベント・ポーリングへの影響』

コネクター用の機能の使用可能化

保証付きイベント・デリバリー機能を使用可能にするには、表 3 に示す値をコネクター構成プロパティーに設定する必要があります。

表 3. JMS 以外のイベント・ストアを使用するコネクターの、保証付きイベント・デリバリー機能関連のコネクター・プロパティー

コネクター・プロパティー	値
DeliveryTransport	JMS
DuplicateEventElimination	true
MonitorQueue	コネクター・フレームワークが、処理済みのビジネス・オブジェクトの ObjectEventId を格納する JMS モニター・キューの名前

保証付きイベント・デリバリーを使用するようにコネクターを構成する場合、表 3 に記載されているコネクター・プロパティーを設定する必要があります。これらのコネクター構成プロパティーを設定するには、Connector Configurator ツールを使用します。このツールを使用すると、これらのコネクター・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。Connector Configurator の詳細については、73 ページの『付録 B. Connector Configurator』を参照してください。

イベント・ポーリングへの影響

DuplicateEventElimination を true に設定して、コネクターで保証付きイベント・デリバリー機能を使用すると、この機能を使用しない場合と比べて、コネクターの動作が多少変化します。重複イベント回避機能を使用するには、コネクター・フレームワークで JMS モニター・キューを使用してビジネス・オブジェクトを追跡します。JMS モニター・キューの名前は、MonitorQueue コネクター構成プロパティーから取得します。

コネクター・フレームワークは、(pollForEvents() メソッドの gotApplEvent() への呼び出しにより) アプリケーション固有のコンポーネントからビジネス・オブジェクトを受け取った後、(gotApplEvents() から受け取った) 現在のビジネス・オブジェクトが重複したイベントを表しているかどうかを判別する必要があります。この判別を行うために、コネクター・フレームワークは JMS モニター・キューからビジネス・オブジェクトを検索し、その ObjectEventId を現在のビジネス・オブジェクトの ObjectEventId と比較します。

- これら 2 つの ObjectEventId が同じであれば、現在のビジネス・オブジェクトが重複イベントであるということになります。このような場合、コネクター・フレームワークは、現在のビジネス・オブジェクトが表すイベントを無視します。つまり、このイベントを統合ブローカーに*送信しません*。
- これらの ObjectEventId が*同じでない* 場合、ビジネス・オブジェクトは重複イベントではありません。この場合、コネクター・フレームワークは、現在のビジネス・オブジェクトを JMS モニター・キューにコピーします。gotApplEvent() メソッドを呼び出した後は、制御はコネクターの pollForEvents() メソッドに戻ります。

重複イベント回避機能をサポートするコネクターの場合は、コネクターの pollForEvents() メソッドで、以下のステップを行う必要があります。

- JMS 以外のイベント・ストアから検索したイベント・レコードからビジネス・オブジェクトを作成した場合は、イベント・レコードの固有イベント ID をビジネス・オブジェクトの ObjectEventId 属性として保管してください。

  アプリケーションは、イベント・ストアのイベント・レコードを一意に識別するため、このイベント ID を生成します。統合ブローカーへイベントを送信してから、このイベント・レコードの状況が変更可能となる前に、コネクターに障害が発生した場合、このイベント・レコードは「進行中」状況のままイベント・ストアに残されます。コネクターが復旧した際に、「進行中」のイベントをリカバリーする必要があります。コネクターは、ポーリングを再開すると、イベント・ストアに残っているイベント・レコードのビジネス・オブジェクトを生成します。ただし、すでに送信済みのビジネス・オブジェクトと新規ビジネス・オブジェクトの両方がそれらの ObjectEventId として同じイベント・レコードを持っているため、コネクター・フレームワークは、新規ビジネス・オブジェクトを重複オブジェクトと認識し、それを統合ブローカーに送信しません。
- コネクターのリカバリー時には、コネクターが新規イベントのためのポーリングを開始する*前*に、「進行中」のイベントを処理するようにしてください。

  コネクターの開始時に、「進行中」のイベントが「ポーリング可能」状況に変更されない限り、ポーリング・メソッドは再処理のためにイベント・レコードを受信しません。

コネクターの複数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、 27 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 4 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 4. コネクターの始動スクリプト

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

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

- Windows システムで**「スタート」**メニューから。

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

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

– UNIX ベースのシステム:

```
connector_manager_connName -start
```

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

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

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

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

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

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

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

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

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

コネクターの初期化

初期化中、コネクターは次のように機能します。

- 構成プロパティーを検索します
- 構成された入力キューと出力キューのアクセス可能性を調べます
- データ・ハンドラー・マッピング・メタデータをロードします

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

- 『アダプターのビジネス・オブジェクトの構造』
- 33 ページの『イベント処理のビジネス・オブジェクト定義生成の概要』
- 37 ページの『エラー処理』
- 38 ページの『トレース』

コネクターに付属しているのは、サンプル・ビジネス・オブジェクトだけです。システム・インテグレーター、コンサルタント、カスタマーは、ビジネス・オブジェクトを構築する必要があります。ビジネス・オブジェクトの構築の詳細については、33 ページの『イベント処理のビジネス・オブジェクト定義生成の概要』を参照してください。

コネクターはメタデータ主導型です。WebSphere Business Integration システムのビジネス・オブジェクトでは、メタデータがアプリケーションに関するデータになります。メタデータは、ビジネス・オブジェクト定義に格納され、コネクターとアプリケーションの連動に利用されます。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・オブジェクトを処理する際に、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を行います。

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ています。コネクターはメタデータ主導型であるため、コネクター・コードを変更する必要なしに、新規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。ただし、コネクターの構成済みデータ・ハンドラーでは、サポートされるビジネス・オブジェクトの構造、オブジェクト・カーディナリティー、アプリケーション固有の情報の形式、およびビジネス・オブジェクトのデータベース表記に関する前提事項が想定されます。したがって、QAD MFG/PRO 向けビジネス・オブジェクトを作成または変更する場合は、コネクターがそれに従うように設計されている規則に準拠して変更を行う必要があります。そうしないと、コネクターは新規のまたは変更されたビジネス・オブジェクトを適切に処理できません。

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、新規のビジネス・オブジェクトをインプリメントするためのガイドとして使用できます。

アダプターのビジネス・オブジェクトの構造

アダプターのインストール後、ビジネス・オブジェクトを作成する必要があります。ビジネス・オブジェクト定義は、QAD 定義の QDoc XML スキーマを基にしています。QDoc は、専有 XML フォーマットです。ビジネス・オブジェクトの構造には、イベント処理に対応するものと要求処理に対応するものの 2 種類が存在します。

© Copyright IBM Corp. 2004

イベント処理のビジネス・オブジェクトの構造

QAD MFG/PRO ビジネス・オブジェクトは、QDoc の構造をミラーリングします。イベント処理の場合、この定義には QDoc ヘッダーと QDoc 本文の第 1 レベルの子ビジネス・オブジェクトが 2 つ含まれています。アダプターをインストールした際、本文のプレースホルダーを含むビジネス・オブジェクト定義を作成するためのヘッダー・ビジネス・オブジェクト定義 (すべての QDocs に共通) とテンプレートもインストールされています。本文のビジネス・オブジェクトは、メッセージ・タイプに固有のものです。イベント処理に関するビジネス・オブジェクト定義作成の指針については、33 ページの『イベント処理のビジネス・オブジェクト定義生成の概要』を参照してください。

ビジネス・オブジェクトの名前は、対応する QDoc スキーマの名前にプレフィックス `BIA_` とサフィックス `BO` を付けたものです。例えば、`BIA_maintainSupplierBO` のようになります。

注: アダプ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。

要求処理のビジネス・オブジェクトの構造

同期処理である要求処理の場合は、トップレベル・オブジェクト (TLO) を作成します。TLO は、要求、応答、障害の子ビジネス・オブジェクトが格納されている特殊オブジェクトです。要求の子 BO は QDoc XML スキーマに対応し、これを基にして生成されます。ここには、ヘッダーと本文の部分があります。データ・ハンドラーは、要求の子ビジネス・オブジェクトを QDoc SOAP/XML メッセージに変換します。次に、コネクターのプロトコル・ハンドラーは、要求メッセージを QXtend Web サービスに送信します。プロトコル・ハンドラーは、応答 QDoc を受信すると、TLO の応答の子 BO に応答データを取り込みます。応答の子 BO は、QDoc XML 応答スキーマに対応しています。障害メッセージが戻ると、コネクターは障害の子 BO にデータを取り込みます。BIA_FaultBO は、SOAP 1.2 スキーマ (http://www.w3.org/2002/12/soap-envelope) を基に、XML ODA を使用して作成されています。

ビジネス・オブジェクト定義には、対応する QDoc の名前が使用され、`QDoc_TLO` サフィックスが付きます。

要求処理に関するビジネス・オブジェクト定義作成の指針については、34 ページの『要求処理の TLO 生成の概要』を参照してください。

イベント処理のビジネス・オブジェクト定義生成の概要

イベント処理のビジネス・オブジェクトは、QDoc スキーマと WebSphere ビジネス・オブジェクト・テンプレートを基にして生成します。QDoc スキーマは、メッセージの内容を記述した XML スキーマです。図 1 に示されているように、自動化されたプロセスで、XML Object Discovery Agent (ODA) と Business Object Designer を使用して QDoc スキーマに準拠したビジネス・オブジェクト (BO) を作成します。

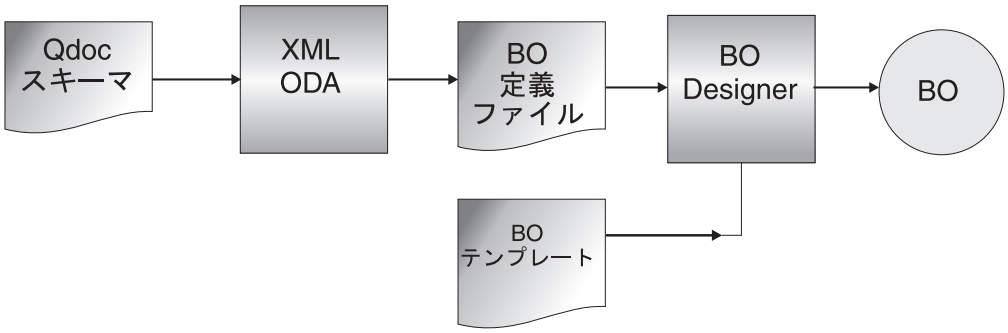

図 7. QDoc スキーマを基にしたイベント処理のビジネス・オブジェクトの生成

XML ODA の実行については、「*データ・ハンドラー・ガイド*」を参照してください。XML ODA への入力データとして QDoc スキーマ (DTD ではなく) を使用します。

XML ODA は、QDoc 全体に対する定義ではなく、メッセージのヘッダーと本文に対する個別の定義だけを生成します。XML ODA は、すべてのメッセージに共通のヘッダー・ビジネス・オブジェクト定義を生成します。この定義 `BIA_QDocHeaderBO` が、アダプターのインストール時に提供されます。また、QDoc テンプレート・ビジネス・オブジェクト定義 `BIA_TemplateEnvelopeBO` が提供されます。このテンプレートを、QDoc ビジネス・オブジェクト定義の作成に使用できます。テンプレートには、QDoc のヘッダーと本文用の 2 つの属性が含まれています。図 8 に、Business Object Designer の `BIA_TemplateEnvelopeBO` を示します。

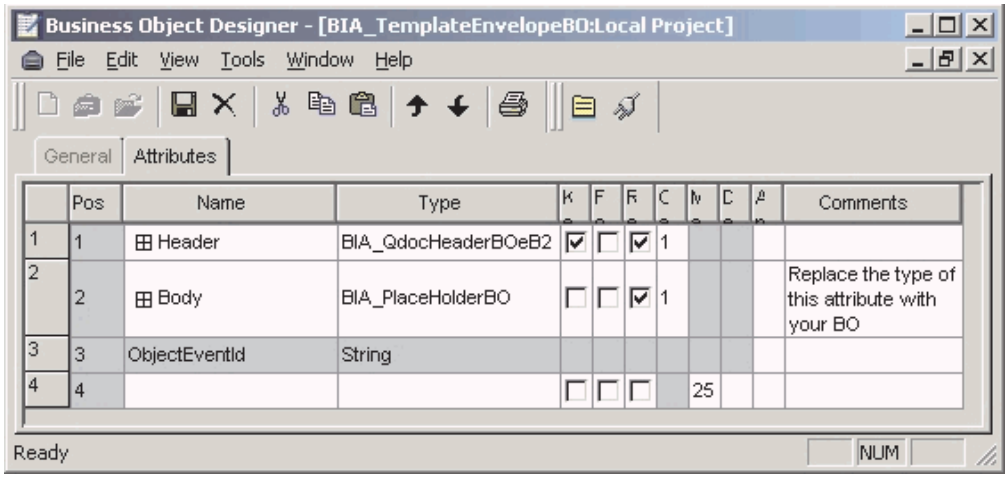

図 8. BIA_TemplateEnvelopeBO

イベント処理のビジネス・オブジェクト定義の生成: ステップバイステップ

イベント処理のために QDoc のビジネス・オブジェクト定義を生成するには、次の手順を実行します。

1. 必要な QDoc スキーマを QAD Web サイトからダウンロードするか、またはすべての .xsd ファイルを *<QXtend のインストール先ディレクトリー>*/webapps/qxtendserver/WEB-INF/schemas/eB2 から *ProductDir*/dependencies/QDocSchemas にコピーして、QDoc スキーマ・ファイルを QXtend 製品から取得します。
2. XML ODA を使用して、QDoc スキーマから QDoc 本文の定義を生成します。例えば、maintainSupplier の場合、QDoc は maintainSupplier.xsd を ODA 用の入力データとして使用し、定義を同じ名前 (maintainSupplier) で保存します。
3. Business Object Designer で BIA_TemplateEnvelopeBO を開きます。

   注: このテンプレートやその他の BO テンプレートは、インストール先ディレクトリー *ProductDir*/connectors/QAD/samples にあります。
4. 本文のタイプを、以前生成した本文の実際の BO 定義 (例えば、図 9に示す maintainCustomer) に変更します。

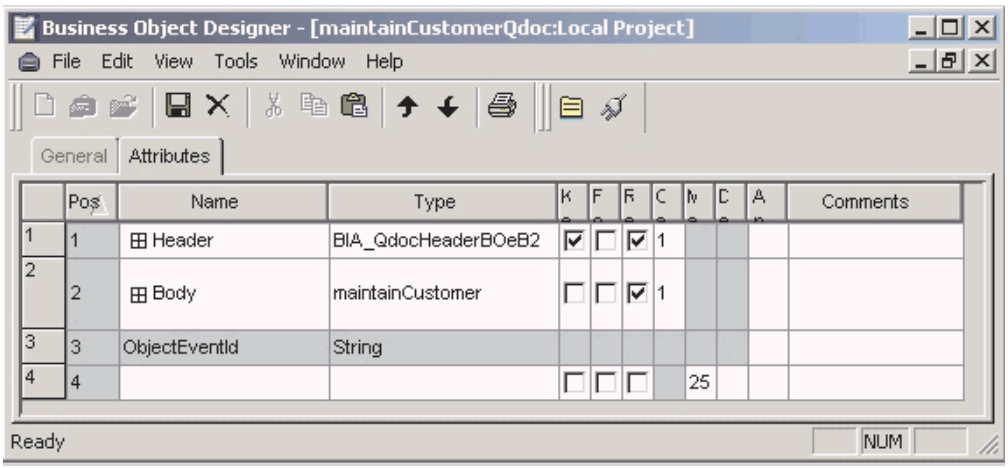

図 9. maintainCustomer の本文が記述された BO エンベロープ

5. 新規ビジネス・オブジェクト定義を *QDoc_name*BO として保存します。ここで、*QDoc_name* は、先頭文字を大文字にした QDoc の名前です。前述の例では、新規ビジネス・オブジェクトの名前は MaintainCustomerBO になります。

要求処理の TLO 生成の概要

要求処理の TLO を作成するには、QDoc スキーマ、ビジネス・オブジェクト・テンプレート、XML ODA、および Business Object Designer を使用します。TLO には、要求、応答、障害の各ビジネス・オブジェクトが格納されています。

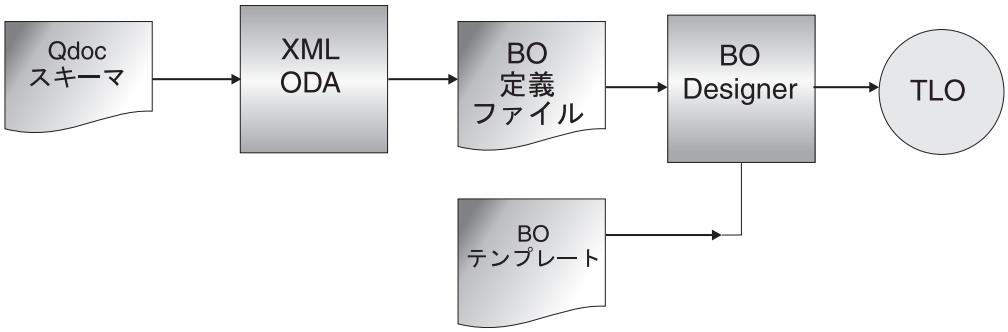

図 10. QDoc スキーマを基にした要求処理の TLO の生成

図 11 に、要求処理に使用する TLO の構造を示します。要求、応答、障害の各属性は、ビジネス・オブジェクトに対応します。

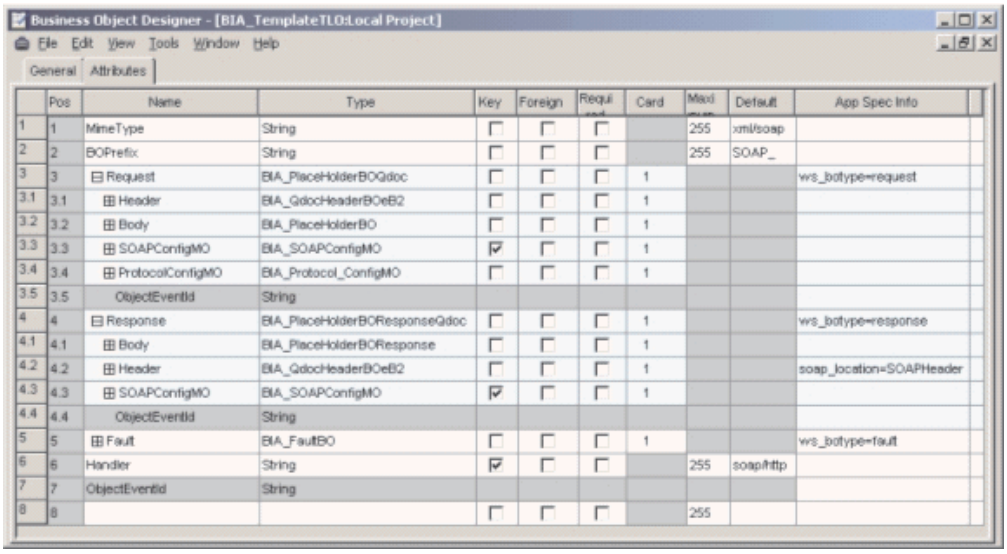

図 11. BIA_TemplateTLO

要求 BO 定義の ProtocolConfigMO には、QXtend Web サービスの URL を指定する Destination 属性が格納されています。この属性の値を目的の QXtend Web サービス URL に設定します。または、QXtend Web サービスのデフォルトの URL を、コネクター固有のプロパティー QXtendDefaultURL に指定することもできます。要求ビジネス・オブジェクトの ProtocolConfigMO の Destination 属性にURL を定義しなかった場合は、QXtendDefaultURL プロパティーの値を使用します。

テンプレート TLO である `BIA_TemplateTLO` が、アダプターによってインストールされます。XML ODA の実行については、「データ・ハンドラー・ガイド」を参照してください。XML ODA への入力データとして QDoc スキーマ (DTD ではなく) を使用します。

要求処理の TLO を生成するには、 36 ページの『要求処理の TLO の生成: ステップバイステップ』を参照してください。

要求処理の TLO の生成: ステップバイステップ

実際の TLO を作成するには、以下の手順を実行します。

1. 必要な QDoc スキーマを QAD Web サイトからダウンロードするか、またはすべての .xsd ファイルを *<QXtend のインストール先ディレクトリー>*/webapps/qxtendserver/WEB-INF/schemas/eB2 から *ProductDir*/dependencies/QDocSchemas にコピーして、QDoc スキーマ・ファイルを QXtend 製品から取得します。
2. XML ODA を使用して、QDoc スキーマから QDoc 本文の定義を生成します。例えば、maintainSupplier の場合、QDoc は maintainSupplier.xsd を ODA 用の入力データとして使用し、定義を同じ名前 (maintainSupplier) で保存します。
3. Business Object Designer で BIA_PlaceHolderBOQDoc を開きます。

   注: このテンプレートやその他の BO テンプレートは、インストール先ディレクトリー *ProductDir*/connectors/QAD/samples にあります。
4. 本文のタイプを、以前生成した本文の実際の BO 定義に変更します。この例では、maintainSupplier になります。
5. 新規の定義を *<qdoc_name>*RequestQDoc として保存します。ここで、*<qdoc_name>* は QDoc の名前です。前述の例では、新規ビジネス・オブジェクトの名前は maintainSupplierRequestQDoc になります。
6. 要求オブジェクトの ProtocolConfigMO を開き、QXtend Web サービスの URL を Destination 属性として指定します。図 12 には、要求 BO の ProtocolConfigMO の Destination 属性の値を示します。

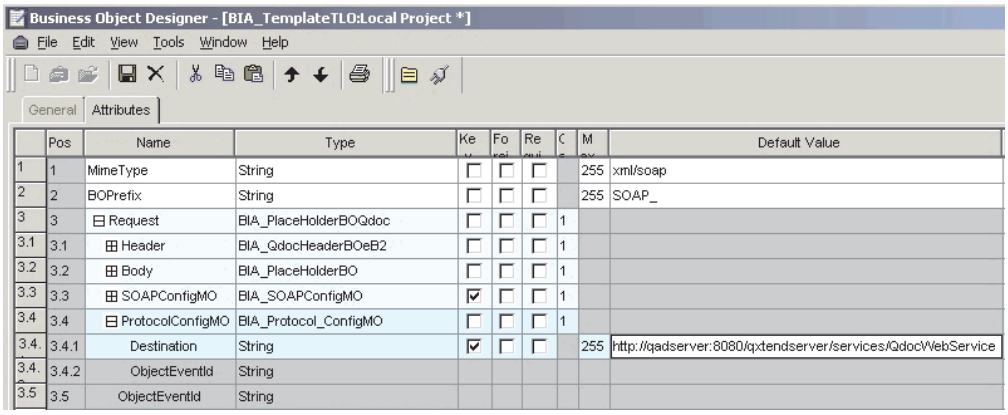

図 12. ProtocolConfigMO Destination 属性

7. XML ODA を使用して、QDoc スキーマから応答 QDoc 本文の定義を生成します。例えば、maintainSupplierResponse の場合、QDoc では ODA の入力として maintainSupplierResponse.xsd を使用します。次に、同じ名前 (maintainSupplierResponse) の下に定義を保存します。
8. Business Object Designer で、BIA_PlaceHolderBOResponseQDoc をロードします。

   注: このテンプレートやその他の BO テンプレートは、インストール先ディレクトリー *ProductDir*/connectors/QAD/samples にあります。

9. 本文のタイプを、以前生成した本文の実際の BO 定義に変更します。この例では、maintainSupplierResponse になります。
10. 新規の定義を *<qdoc_name>*ResponseQDoc として保存します。ここで、*<qdoc_name>* は QDoc の名前です。前述の例では、新規ビジネス・オブジェクトの名前は maintainSupplierResponseQDoc になります。
11. Business Object Designer で BIA_TemplateTLO を開きます。

    注: このテンプレートやその他の BO テンプレートは、インストール先ディレクトリー *ProductDir*/connectors/QAD/samples にあります。
12. 要求タイプを、BIA_PlaceHolderBOQDoc から、要求 BO に対して以前保存したタイプに変更します。前述の例では、名前は maintainSupplierRequestQDoc になります。
13. 応答タイプを、BIA_PlaceHolderBOResponseQDoc から、応答 BO に対して以前保存したタイプに変更します。前述の例では、名前は maintainSupplierResponseQDoc になります。
14. 新規の定義を *<qdoc_name>*QDoc TLO として保存します。ここで、*<qdoc_name>* は QDoc の名前です。前述の例では、新規 TLO の名前は maintainSupplierQDoc_TLO になります。

エラー処理

コネクターによって生成されるエラー・メッセージはすべて、BIA_QADConnector.txt という名前のメッセージ・ファイルに保管されます。(ファイルの名前は、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、個別のエラーを以下の各セクションで説明するように処理します。

イベント処理

- イベント・キューにアクセスできない場合、コネクターは致命的エラーをログに記録して終了します。
- データ・ハンドラーがビジネス・オブジェクトを生成できない場合、コネクターはエラーをログに記録します。詳細については、41 ページの『第 4 章 QAD MFG/PRO データ・ハンドラー』を参照してください。
- ビジネス・オブジェクトがサブスクライブされていない場合、コネクターはエラーをログに記録します。詳細については、42 ページの『QAD MFG/PRO データ・ハンドラーの構成』を参照してください。
- ビジネス・オブジェクトの送達が失敗した場合、コネクターはエラーをログに記録し、キュー内にメッセージを残します。

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た。
- 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。

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、コネクターの動作を詳細に追跡するために使用できるオプショナル・デバッグ機能です。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、15 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。トレースを使用可能にして設定する方法などのトレースの詳細については、「*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　このレベルは、ビジネス・オブジェクトが統合ブローカーにポストされたときや要求ビジネス・オブジェクトが受信されたときにそれを記録するトレース・メッセージ、またはコネクターが処理する各オブジェクト用のビジネス・オブジェクト・ハンドラーを示すトレース・メッセージに使用されます。

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　このレベルは、コネクターが機能を始動または終了したときにそれを示すトレース・メッセージ、アプリケーション固有の情報 (例えば、ビジネス・オブジェクト内の ASI フィールドを処理するメソッドが戻す値など) を示すトレース・メッセージ、またはスレッド固有の処理を記録するトレース・メッセージ (例えば、コネクターが複数のスレッドを生成すると、トレース・メッセージは新しいスレッドが作成されるたびにそれを記録します) に使用されます。

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。

第 4 章 QAD MFG/PRO データ・ハンドラー

QAD MFG/PRO データ・ハンドラーは、主にビジネス・オブジェクトを MFG/PRO フォーマットに変換する役割を担うデータ変換モジュールです。これらのフォーマットの概要については、 7 ページの『データ・フォーマット』を参照してください。 この章では、データ・ハンドラーがメッセージをあるフォーマットから別のフォーマットへ処理する仕組みと、メタデータ要件について説明します。

イベント処理: 図 13 に示されているように、QAD MFG/PRO データ・ハンドラーに備わったマッピング・エンジンによって、トリプレット・メッセージが、QAD の専有 XML フォーマットである XML QDoc に変換されます。次に、コネクターの XML データ・ハンドラーが起動し、QDoc が対応するビジネス・オブジェクト (BO) に変換されます。イベント・メッセージが XML フォーマットで到着すると、コネクターはトリプレットから QDoc への変換処理をスキップし、メッセージを XML データ・ハンドラーに渡してイベント・ビジネス・オブジェクトへの変換に備えます。(ビジネス・オブジェクトは、設計時に QDoc スキーマから生成されます。詳細については、 33 ページの『イベント処理のビジネス・オブジェクト定義生成の概要』を参照してください。) データ・ハンドラーは、QDoc でオペレーション・タグの値を適用し、ビジネス・オブジェクト・レベルで、また、子ビジネス・オブジェクトに対して、動詞を設定します。

メタデータは、フォーマット変換で使用されます。

- QADDataHandlerMO (メタオブジェクト) で指定されたメタデータは、このメタオブジェクトで、トリプレットから QDoc へのマッピングに使用されます (メタオブジェクトの詳細については、 42 ページの『QAD MFG/PRO データ・ハンドラーの構成』を参照してください)。
- QADDataHandlerMO の子 (OutboundMappingMO) オブジェクトに格納されたメタデータは、QDoc から BO へのマッピングに使用されます。

© Copyright IBM Corp. 2004

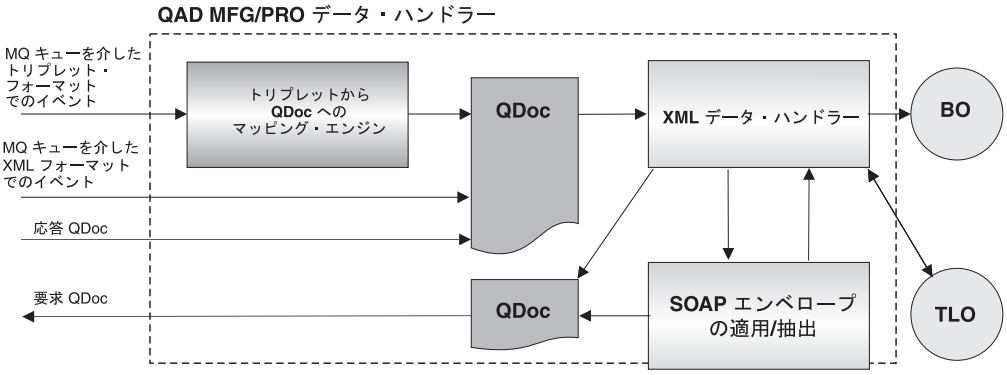

図 13. QAD MFG/PRO データ・ハンドラー処理

要求処理 反対方向では、XML データ・ハンドラー・コンポーネントが TLO の要求ビジネス・オブジェクトを QDoc XML メッセージに変換します。このメッセージは、次に SOAP 1.2 エンベロープにラップされます。要求 BO には、BIA_QdocHeaderBOeB2 というタイプのヘッダー属性と、要求に固有の Body 属性が必要です。プロトコル・ハンドラーは、その後 QDoc を QXtend Web サービスに渡します。

応答 QDoc を受信すると、コネクターは XML データ・ハンドラーを呼び出して、応答ビジネス・オブジェクトを作成し、SOAP エンベロープを抽出します。要求 BO と同様に、応答 BO には BIA_QdocHeaderBOeB2 というタイプのヘッダーと Body 属性があります。障害 QDoc メッセージが戻った場合は、コネクターがデータ・ハンドラーを呼び出して、障害 QDoc を BIA_FaultBO に変換します。障害メッセージでは、QDoc ヘッダーは廃棄されます。

『QAD MFG/PRO データ・ハンドラーの構成』で説明するように、メタデータには要求処理の変換を誘導する機能があります。

QAD MFG/PRO データ・ハンドラーの構成

QAD MFG/PRO データ・ハンドラーは、QAD MFG/PRO のコネクターの中心的なコンポーネントです。コネクターはデータ・ハンドラーを呼び出して、ビジネス・オブジェクトを QDoc メッセージに変換し、トリプレット・メッセージおよび QDoc メッセージをビジネス・オブジェクトに変換します。

この変換で、データ・ハンドラーのメタオブジェクト内の情報が重要な役割を果たします。製品ファイルをインストールしたら、始動する前に、この情報を構成してください。QAD MFG/PRO データ・ハンドラーをカスタマイズまたは拡張しない場合は、次の項目のみ構成してください。

- Connector Configurator を使用して、コネクター固有プロパティー DataHandlerMetaObjectName の値を BIA_QADDataHandlerMO に設定します。Connector Configurator の起動と使用方法、およびその手順については、73 ページの『付録 B. Connector Configurator』を参照してください。

43 ページの図 14 に、BIA_QADDataHandlerMO および OutboundMappingMO オブジェクトを含む、データ・ハンドラーのメタオブジェクト階層を示します。

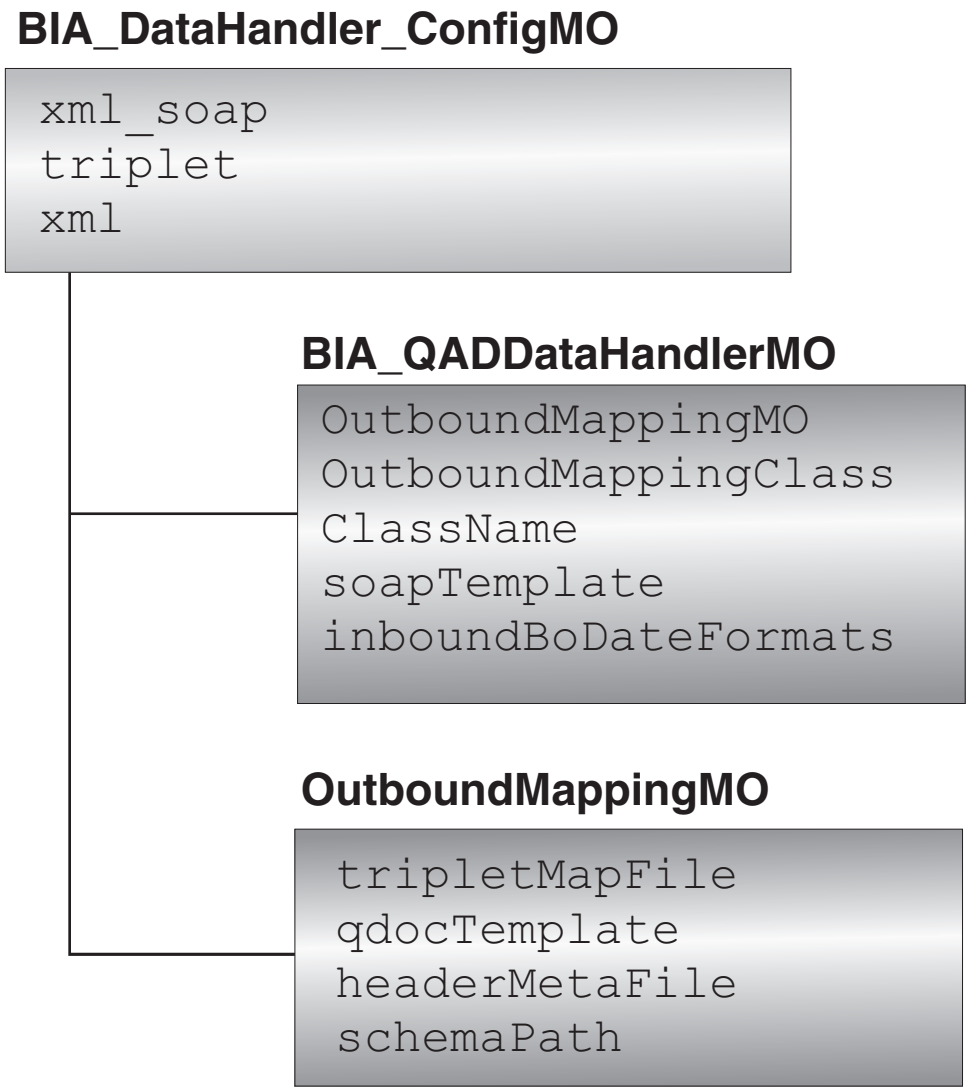

図 14. データ・ハンドラーのメタオブジェクト

表 5 には、BIA_DataHandlerConfigMO に必要な属性が説明されています。このトップレベル・メタオブジェクトの名前は、DataHandlerMetaObjectName コネクター構成プロパティーから読み取ります。

表 5. BIA_DataHandlerConfigMO のメタオブジェクト属性

属性名	タイプ	説明
xml_soap	BIA_QADDataHandlerMO	xml_soap MIME タイプと組み合わせて使用する、QAD MFG/PRO データ・ハンドラーのメタオブジェクト
triplet	BIA_QADDataHandlerMO	triplet MIME タイプと組み合わせて使用する、QAD MFG/PRO データ・ハンドラーのメタオブジェクト
xml	BIA_XMLDataHandlerMO	QAD MFG/PRO データ・ハンドラーのコンポーネントとして構成される、XML データ・ハンドラーのメタオブジェクト

表 6 で、QADDataHandlerMO の属性について説明します。

表 6. QADDataHandlerMO のメタオブジェクトの属性

属性名	タイプ	説明
OutboundMappingMO	String	アウトバウンド・マッピング用のメタオブジェクトの名前 (イベント処理)。この属性が CxIgnore に設定されている場合、アウトバウンド・マッピングは実行されません。
OutboundMappingClass	String	イベント処理マッピング・ハンドラーのクラス名。このクラスは、トリプレットから QDoc XML へのマッピングに使用されます。この属性が CxIgnore に設定されている場合、トリプレットから Qdoc へのマッピングは実行されません。
ClassName	String	com.ibm.adapters.qad.datahandlers.QADDataHandler という値を持つデータ・ハンドラー・クラス。この値は変更しないでください。
soapTemplate	String	SOAP エンベロープ・テンプレート・ファイル BIA_SoapEnvelopeTemplate.xml の完全修飾パス名。
inboundBoDateFormats	String	着信日付フィールドの解析に使用するパターンを「;」で区切ったリスト。デフォルト: mm/dd/yyyy;mm-dd-yyyy;yyyy-mm-dd

表 7 で、OutboundMappingMO (イベント処理) の属性について説明します。

表 7. OutboundMappingMO (イベント処理) のメタオブジェクトの属性

属性名	タイプ	説明
tripletMapFile	String	トリプレット XML マッピング・ファイルの完全修飾パス名
qdocTemplate	String	QDoc テンプレート・ファイルの完全修飾パス名
headerMetaFile	String	トリプレットから QDoc へのヘッダー・マッピングを含むファイルの完全修飾パス名
schemaPath	String	QDoc XML スキーマ・ファイルを含むルート・ディレクトリーの完全修飾パス名

BIA_XMLDataHandlerMO 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。マッピング・メタデータと、フォーマット変換におけるその役割については、以降の各セクションで説明します。

QAD MFG/PRO データ・ハンドラー処理

QAD MFG/PRO データ・ハンドラーは、次のように、QAD MFG/PRO メッセージとビジネス・オブジェクトの間で変換を実行します。

- **MFG/PRO メッセージからビジネス・オブジェクトへの処理**
  - QAD MFG/PRO メッセージからビジネス・オブジェクトへのデータ処理は、QAD MFG/PRO アプリケーションが WebSphere ビジネス・プロセスを呼び出す際のイベント処理中に行われます。データ・ハンドラーは、まずトリプレッ

ト・メッセージを QDoc に変換するマッピング・エンジンを呼び出し、次に XML データ・ハンドラーを起動して QDoc をビジネス・オブジェクトに変換します。メッセージがトリプレットではなく QDoc である場合、マッピング・エンジンはバイパスされ、XML データ・ハンドラーが直接呼び出されます。

- **ビジネス・オブジェクトから MFG/PRO メッセージへの処理**は、要求の処理中に実行されます。データ・ハンドラーは、XML データ・ハンドラーを使用して TLO の要求ビジネス・オブジェクトを要求 QDoc に変換し、SOAP 1.2 エンベロープのメッセージをラップします。

QAD MFG/PRO メッセージからビジネス・オブジェクトへの処理

各セクションの内容は、以下のとおりです。

- トリプレットから QDoc への処理
  - 規則
  - メタデータ
  - 処理の手順
  - ヘッダーのマッピング
- QDoc からビジネス・オブジェクトへの処理
- このようなトランザクションに必要なマッピング・メタデータ

トリプレットから QDoc への処理: 規則

次の規則は、QAD MFG/PRO データ・ハンドラーのマッピング・エンジンによってトリプレット・メッセージが QDoc XML に変換される場合に適用されます。

- **フィールド要素のマッピング** トリプレット・フィールド名は XML 要素にマップされます。
- **命名規則** 下線で区切られた MFG/PRO 名を英大文字小文字混合の表記に変換することによって (例えば、`pt_pl_line` は `ptPlLine` になります)、トリプレット・メッセージ内のフィールドに対応する要素名をすべて派生できます。
- **名前フィールド関連** トリプレット・フィールドを含む MFG/PRO ビジネス・オブジェクトまたはデータベース・レコードに対応するエンティティ・レベルの要素名は、既存の MFG/PRO メタデータから自動的には派生されません。トリプレット・フィールドは QAD によって設定され、共通のビジネスおよび MFG/PRO の使用法を反映します。例えば、トリプレット内のアクション・フィールドに対応するオペレーション要素は、パブリッシュされたイベントの一部として MFG/PRO 内で更新されたすべての XML 要素上に存在している必要があります。それ自体は影響を受けないが影響を受ける子を参照するためだけに QDoc 内に存在する親要素の場合、オペレーション・フィールドは省略することができます (ただし、子では必須です)。この要素が関係するのは、作成、更新、削除の各要求のみであり、注文の出荷、在庫表の発行、受領などのビジネス・トランザクションには関係しません。
- **配列の表記** QDocs における配列の表記は、SOAP メッセージ内の部分的な配列の送信を禁止する SOAP 1.2 エンコード規格に準拠している必要があります。未記入の配列エントリーと値が空ストリングに設定されている配列エントリーを区別するために、2 つの QDoc 固有の属性「index」および「skip」が定義されています。例えば、次のようになります。

```
<orderQuantity
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:enc="http://www.w3.org/2002/12/soap-encoding"
   xmlns:qcom="http://www.qad.com/qdoc/common"
   enc:itemType="xsi:decimal"
   enc:arraySize="6"
   >
   <!-- Entries 1, 3 are set to empty and 4, 6 are skipped -->
   <qcom:entry index="1"/>
   <qcom:entry index="2">10.583</qcom:entry>
   <qcom:entry index="3"/>
   <qcom:entry index="4" skip=ītrueî/>
   <qcom:entry index="5">22</qcom:entry>
   <qcom:entry index="6" skip="true">ignore!</qcom:entry>
</orderQuantity>
```

- **QDoc の親子テーブルの正規化解除** トリプレット文書には、親子関連によって他のテーブルと密接に関連したテーブルのデータが含まれている場合があります。ほとんどの場合、すべての子テーブルにも影響を与えるイベントのパブリッシュには、親オブジェクトを表す単一の QDoc が使用されます。単一のタイプの QDoc を複数のタイプのトリプレット文書に使用する場合、トリプレット文書の複数インスタンスが実行時にマージされて単一の QDoc インスタンスを生成するわけではありません。一括バッチでイベント・ドリブンである DataSync の性質を保持するには、各トリプレット文書を単一の QDoc にマップします。ただし、同一の QDoc タイプが複数のトリプレット文書タイプによって共用されます。例えば、Analysis Code Master (an_mstr) テーブル、Analysis Code Link Detail (anl_det) テーブル、および Analysis Code Selection Detail (ans_det) テーブルのトリプレット文書タイプが、単一の maintainAnalysisCode QDoc タイプにマップされます。
  マッピングを完了するために、親テーブルのトリプレット・データが、QDoc 内の最高レベルの XML 要素にマップされます。子テーブル・トリプレットのデータは、低レベルの子要素にマップされます。ただし、QDoc XML データは、共通キーの要素が親の中に入れられ、子要素からは除去された状態で、構造化され正規化解除された形式で表されます。XML では、要素の構造的なネスティングで親子関係が暗黙の状態になっているので、こうした冗長な外部キーは必要ありません。マッピング・アルゴリズムは、実行時に各 MFG/PRO テーブルの共通キー・フィールドを認識します。トリプレット文書が親ではなく子のテーブル・レコードを更新すると、その結果の QDoc で、共通キー値は子 XML 要素から親 XML 要素へ移動されます。子テーブル・レベルのトリプレットに表示される共通キーを表すフィールドのマッピングは、子テーブルのプレフィックスと親テーブルのプレフィックスに置き換えることによって実行されます。MFG/PRO 規則により、すべてのフィールド名は、データベース・テーブル名の省略形であるプレフィックス、下線文字、下線文字区切りの 1 つ以上のトークンの順で表され、これら全体によりフィールドが識別されます。親テーブルと子テーブル間の共通キーは、テーブル固有のプレフィックス以外は、必ず同じフィールド名を持ちます。例えば、spt_det 外部キー・フィールドに相当する costSimulation (sct_det) フィールドは、spt_sim の場合は sct_sim、spt_part の場合は sct_part、spt_site の場合は sct_site です。
- **文書からレコードへのマッピング** どのトリプレット文書にも、MFG/PRO データベース・レコード 1 つだけのデータが含まれています。
- **文書からレコードへのマッピングの例外: アドレス・データのあるカスタマーとサプライヤー** カスタマーとサプライヤーのアドレスと同じアドレス・レコードを

再利用できるように、カスタマーとサプライヤーのアドレス・データは、共通の多目的 MFG/PRO テーブル (ad_mstr) および相互参照できる別の場所 (ls_mstr) に保管されます。カスタマーまたはサプライヤーの 1 次アドレスに加えて、特定のカスタマーに関連するいくつかの配送先アドレスや特定のサプライヤーに関連するいくつかの送金先アドレスがあります。カスタマーのトリプレット文書 (cm_mstr)、サプライヤーのトリプレット文書 (vd_mstr)、アドレスのトリプレット文書 (ad_mstr) のテーブルには、カスタマーまたはサプライヤーのマスター・データと混合したアドレス・データが含まれています。そのため、カスタマー・データとサプライヤー・データのトリプレット・フォーマットは、次のように、他のトリプレット文書タイプとはやや異なります。

- データは、改行で区切られたいくつかの行から構成されます。各物理 MFG/PRO データベース・レコードから取り出されたデータは、それぞれ独自の行に保管されます。
- 各行は大括弧で囲まれたテーブル名で始まるのではなく、トークン customer または supplier が cm_mstr または vd_mstr の代わりに使用されます。アドレス・データを含む行は、テーブル名 ad_mstr ではなく、大括弧で囲まれたトークン customeraddress、supplieraddress、ship-toaddress、または remit-toaddress で始まります。どのトークンが使用されるかは、存在するアドレス・データのタイプによって決まります。
- カスタマーまたはサプライヤーの 1 次アドレス・データが更新されると、カスタマーまたはサプライヤーのマスター (cm_mstr または vd_mstr) 要素の固定セットが、保守イベントによる影響を受けていない場合でも、新規アドレス・データとともにパブリッシュされます。
- 「配送先」または「送金先」アドレスが更新されると、カスタマーまたはサプライヤーのマスター (cm_mstr または vd_mstr) 要素の固定セットと、場合によってはカスタマーまたはサプライヤーの 1 次アドレス (ad_mstr) 要素の固定セットが、保守イベントによる影響を受けていない場合でも、新規アドレス・データとともにパブリッシュされます。

トリプレットから QDoc への処理: メタデータ

トリプレットから QDoc への処理に関する前述の規則は、実行時に適用され、マッピング・メタデータによってガイドされます。マッピング・ロジックは、QADDataHandlerMO の子属性でもある OutboundMappingClass にあります。メタデータは、OutboundMappingMO の子である mapFile にあります。mapFile は、メタデータを含む XML ファイルであり、次のように機能します。

- トリプレット・テーブル名を対応する QDoc スキーマおよびビジネス・オブジェクト名にマップします。
- 親ビジネス・オブジェクト名と、子表記で使用される外部キーを指定します。
- また、例外として次の機能があります。
  - テーブル名の代わりに使用されるトリプレット・トークンを、対応する MFG/PRO テーブルの名前にマップします。カスタマーおよびサプライヤーの 1 次アドレス・フィールド (QDoc 内のカスタマーおよびサプライヤーの要素と同じレベルに表示される) を、アドレス・テーブルではなくカスタマーまたはサプライヤーのマスター・テーブルにマップします。

表 8 では、mapFile に保管されている情報について説明します。

表 8. mapFile メタデータ

テーブル名	QDoc 名	BO 名	親 オブジェクト	外部キー
ac_mstr	maintainAccount	account		
an_mstr	maintainAnalysisCode	analysisCode		
anl_det	maintainAnalysisCode	analysisCodeLink	analysisCode	anl_type、 anl_code
ans_det	maintainAnalysisCode	analysisCodeSelection	analysisCode	ans_type、 ans_code
bom_mstr	maintainBillOfMaterial	billOfMaterial		
cc_mstr	maintainCostCenter	costCenter		
ccd1_det	maintainCostCenter	costCenterAcct Validation	costCenter	ccd1_cc
ccd2_det	maintainCostCenter	costCenterSubacct Validation	costCenter	ccd2_cc
cd_det	maintainMasterComments	masterComment		
cm_mstr (関連する ad_mstr データとともにパブリッシュされます)	maintainCustomer	customer		
ad_mstr (カスタマーとサプライヤーの同期化にも利用されます)	maintainAddress または maintainCustomer または maintainSupplier			
ls_mstr (カスタマーとサプライヤーの同期化にも利用され、MFG/PRO 以外のアドレス・タイプに対してのみ独立して同期されます)	maintainAddressList	addressList		
code_mstr	maintainCode	code		
cp_mstr	maintainCustomerItem	item		
cs_mstr	maintainCostSet	costSet		
cu_mstr	maintainCurrency	currency		
dpt_mstr	maintainDepartment	department		
en_mstr	maintainEntity	entity		
exr_rate	maintainExchangeRate	exchangeRate		
fcs_sum	maintainForecastSummary	forecastSummary		
glc_cal	maintainGeneralLedger Calendar	generalLedger Calendar		
is_mstr	maintainInventoryStatus	inventoryStatus		
isd_det	maintainInventoryStatus	inventoryStatus		inventorySta tus
pc_mstr	maintainPurchasingPrice List	purchasingPriceList		
pi_mstr	maintainPriceList	priceList		

表 8. *mapFile メタデータ (続き)*

テーブル名	QDoc 名	BO 名	親 オブジェクト	外部キー
pid_det	maintainPriceList	priceListDetail		priceList
pl_mstr	maintainProductLine	productLine		
ps_mstr	maintainProductStructure	productStructure		
pt_mstr	maintainItem	item		
ro_det	maintainRoutingOperation	routingOperation		
sb_mstr	maintainSubAccount	subAccount		
sbd_det	maintainSubAccount	subAccountDetail		subAccount
si_mstr	maintainSite	site		
sct_det	maintainCostSimulation	costSimulation		
spt_det	maintainCostSimulation	costSimulationItem		costSimulation
um_mstr	maintainAlternateUnitOfMeasure	alternateUnitOfMeasure		
vd_mstr (関連する ad_mstr データとともにパブリッシュされます)	maintainSupplier	supplier		
vp_mstr	maintainSupplierItem	supplierItem		
wc_mstr	maintainWorkCenter	workCenter		

次のサンプル・フラグメントは、mapFile 内のメタデータがどのように項目をマップするかを示しています。

```
<!-- Master table record - no parent -->
<record  name="ac_mstr">
        <qdoc name ="maintainAccount" version="1.0"/>
        <bo> account</bo>
</record>

<!-- child table record ñ parent and foreign keys required -->
<record name= "anl_det" >
        <qdoc name = "maintainAnalysisCode" />
        <bo> analysisCodeLink </bo>
         <parent>an_mstr</parent>
        <fkey>anl_type</fkey>
        <fkey>anl_code</fkey>
</record>

<!-- Customer Master table record - no parent -->
<record name= "cm_mstr">
        <qdoc name="maintainCustomer" />
        <bo> customer </bo>
</record>
<!-- Address master table record -->
<record name= "ad_mstr" >
        <qdoc name="maintainAddress" version="1.1"/>
        <bo>address</bo>
</record>

<!-- exception cases mapping -->
< record name= "customer" >
        <qdoc name="maintainCustomer" />
```

```
        <bo> customer </bo>
</ record >
< record name= "customeraddress"alias="ad_mstr" >
        <qdoc name="maintainCustomer" />
         <parent>customer</parent>
</ record >
< record name= "ship-toaddress" >
        <qdoc name="maintainCustomer" />
         <parent>customer</parent>
        <bo>shipToAddress</bo>
</record>

<exit_on_error>
        <bo> customer</bo>
        <bo> analysisCodeLink </bo>
</exit_on_error>
```

フラグメント内の例外ケースは、通常のレコードと同じ構文を使用してマップされます。ただし、テーブル名の代わりにトリプレット・トークンが指定される点が異なります。alias 属性には、実際のテーブル名が組み込まれ、対応する QDoc 要素名の形成に使用されるプレフィックスを取得するために使用されます。カスタマーおよびサプライヤーの 1 次アドレスの場合、ビジネス・オブジェクト名は指定されません。これにより、アドレス・フィールドは、カスタマー・フィールドとサプライヤー・フィールド内で同じレベルで表示されます。

exit_on_error タグは、指定されたビジネス・オブジェクトの処理中にエラーが発生した場合に、コネクターがシャットダウンするかどうかを示します。デフォルトでは、コネクターはエラーを無視し、イベントの検索と処理を続行します。

トリプレットから QDoc へのヘッダー・マッピング

表 9 に、トリプレット・メッセージ・ヘッダーでの、QDoc ヘッダー要素から QAD MFG/PRO コントロール・タグへのマッピングを示します。

表 9. QDoc とトリプレットのヘッダー・マッピング

QDoc 要素	説明	トリプレットの内容
senderId	送信側の URI	@SYSID タグ
receiverId	受信側の URI	@TRADPTRID タグ
senderDocumentId	QAD MFG/PRO の内部で割り当てられた文書 ID	QDocs 上の senderDocumentId は、QAD MFG/PRO アプリケーションがオリジナル QDoc を相互参照および確認できるように、QDoc Confirmation メッセージ上で参照される必要があります。
descriptor	メッセージに含まれている文書のタイプについて説明したテキスト・ストリングで、複数回、出現します。最初に現れる文字: 文書規格、2 番目に現れる文字: 文書タイプ、3 番目に現れる文字: 文書改訂	使用されません。
confirmationLevel	どのような場合に QDoc Confirmation 文書が要求されるかを指定します。使用可能な値は none、error、または all です。	@ACKLVLREQD タグ
dateTimeCreated	XML スキーマで既定された DateTime タイプで、文書作成の日付と時刻のスタンプが表示されます。	@DATECREATE、@TIMECREATE、および @TIMEZONE のトリプレット・ヘッダー・タグの組み合わせ
senderDocumentRef	複数のストリングの出現 (最大 2 回) で、送信側アプリケーションへの参照が示されます。	@MFGPROSITE (指定されている場合) @MFGPROKEY
receiverDocumentRef	複数のストリングの出現 (最大 2 回) で、受信側アプリケーションへの参照が示されます。	@MFGPROSITE (指定されている場合) @MFGPROKEY

このマッピング・データは、アウトバウンド・マッピング・メタオブジェクトで参照されるテキスト・ファイルに保管されます。ファイルの内容は以下のとおりです。

```
senderId @SYSID
receiverId @TRADPTRID
senderDocumentId @DOCID
confirmationLevel @ACKLVLREQD
dateTimeCreated @DATECREATE:@TIMECREATE:@TIMEZONE
senderDocumentRef @MFGPROSITE:@MFGPROKEY
receiverDocumentRef @MFGPROSITE:@MFGPROKEY
```

QDoc からビジネス・オブジェクトへの処理

QDoc が検出されると、QAD MFG/PRO データ・ハンドラーは XML データ・ハンドラーを起動して、対応するビジネス・オブジェクトを作成します。ビジネス・オブジェクトの動詞は、QDoc 内で対応するオペレーション・タグの値に基づいています。

- QDoc 内のルート・エンティティーでオペレーション・タグが設定されている場合、ビジネス・オブジェクトの動詞は次のように設定されます。
  - A の場合は Create
  - C の場合は Update

– R の場合は Delete

- QDoc 内のルート・エンティティーでオペレーション・タグが設定されていない場合は、子エンティティーからオペレーション値が取り出され、次のように処理されます。
  1. ルート・ビジネス・オブジェクトの動詞が <action>Child に設定されます。ここで、<action> はオペレーション値に基づいて決まります。つまり、A の場合は Create、C の場合は Update、R の場合は Delete になります。
  2. 子ビジネス・オブジェクトの動詞が、オペレーションに適切な値に設定されます。

XML データ・ハンドラー処理の詳細については、「データ・ハンドラー・ガイド」を参照してください。

ビジネス・オブジェクトから QAD MFG/PRO へのメッセージ処理

要求処理の場合、コネクターは TLO の要求の部分を QAD MFG/PRO データ・ハンドラーに渡します。データ・ハンドラーは XML データ・ハンドラーを呼び出して、要求のヘッダーと本文の部分を XML に変換します。XML データ・ハンドラーは、これらの断片を組み立てて SOAP-XML 1.2 メッセージを構築し、コネクターに戻します。コネクターは QXtend Web サービスに対して HTTP 要求を実行し、応答を待ちます。応答メッセージは QAD MFG/PRO データ・ハンドラーによって BO に変換されます。QAD MFG/PRO データ・ハンドラーは、作業の一部を実行するために XML データ・ハンドラーをもう一度呼び出します。このビジネス・オブジェクトは TLO の応答属性に書き込まれ、この TLO がブローカーに戻ります。QXtend Web サービスが、応答 QDoc メッセージではなく障害 QDoc メッセージを戻した場合、QAD MFG/PRO データ・ハンドラーはこれを BIA_FaultBO ビジネス・オブジェクトに変換します。プロトコル・ハンドラーはこのビジネス・オブジェクトを TLO の障害属性の値として設定し、ヘッダーは除去されます。

第 5 章 トラブルシューティング

この章では、コネクターの始動または稼働時に発生する可能性のある問題について説明します。

始動時の問題

アダプターに予期しない障害が発生した場合は、AgentTraceLevel コネクター構成プロパティーでトレース・レベル 5 を指定して、再始動します。トレースの詳細については、38 ページの『トレース』を参照してください。AgentTraceLevel プロパティーの詳細については、55 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。次の表に、その他の始動時のトラブルシューティングを示します。

問題	考えられる解決策と説明
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。Windows の場合: システムの環境変数 %MQ_LIB% が IBM WebSphere MQ Java クライアント・ライブラリーのフォルダーを指していることを確認してください。Unix の場合: ファイル <wbia>/bin/CWSharedEnv.sh の変数 $MQ_LIB が、IBM WebSphere MQ Java クライアント・ライブラリーのフォルダーに設定されていることを確認してください。
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。 Windows の場合: システムの環境変数 %MQ_LIB% が IBM WebSphere MQ Java クライアント・ライブラリーのフォルダーを指していることを確認してください。 Unix の場合: ファイル <wbia>/bin/CWSharedEnv.sh の変数 $MQ_LIB が IBM WebSphere MQ Java クライアント・ライブラリーのフォルダーに設定されていることを確認してください。
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。Windows の場合: システムの環境変数 %MQ_LIB% が IBM WebSphere MQ Java クライアント・ライブラリーのフォルダーを指していることを確認してください。 Unix の場合: ファイル <wbia>/bin/CWSharedEnv.sh の変数 $MQ_LIB が IBM WebSphere MQ Java クライアント・ライブラリーのフォルダーに設定されていることを確認してください。

© Copyright IBM Corp. 2004

問題	考えられる解決策と説明
初期化中にコネクターが予期せずシャットダウンし、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。
コネクターが「MQJMS2005: failed to create MQQueueManager for ':'」を報告する。	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

コネクターが予期せず終了した場合は、InProgressQueue キューに未処理のイベントがないか調べてください。InProgressQueue の詳細については、21 ページの『InProgressQueue』を参照してください。

QAD MFG/PRO アプリケーションへの接続の切断

イベント処理の場合、アダプターは WebSphere MQ キューを介してアプリケーションと通信します。キュー・マネージャーが起動し稼働していることを確認してください。要求処理の場合は、SOAP-HTTP が通信に使用されます。Qxtend のインストール先である Tomcat アプリケーション・サーバーが起動し稼働中であることと、アダプター・マシンからこのマシンに到達できることを確認してください。このことをテストするには、アダプター・マシンでブラウザーを開き、URL http://*qxtendmachine*:8080/qxtendserver を開きます。ここで、*qxtendmachine* は Tomcat が稼働しているマシンのホスト名です。8080 が有効なのは、Tomcat でデフォルト値を変更していない場合に限ります。この URL が開かない場合やエラー・ページの情報が表示された場合は、Qxtend が稼働していないか、または Qxtend に到達できないことを意味しています。詳細については、Qxtend の資料を参照してください。

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 アダプターのコネクター・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、次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、そのブローカーで実行されるアダプターについて構成する必要のある標準プロパティーのリストが表示されます。

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

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

新規プロパティーと削除されたプロパティー

本リリースには、次の標準プロパティーが追加されました。

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

アダプター・コネクターには 2 つのタイプの構成プロパティーがあります。

- 標準構成プロパティー
- コネクター固有のプロパティー

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

© Copyright IBM Corp. 2004

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、付録 B『Connector Configurator』を参照してください。

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

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

標準プロパティーの要約

表 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。

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>
AgentTraceLevel	0 から 5	0	動的	
ApplicationName	アプリケーション名	コネクター・アプリケーション名として指定された値	コンポーネント再始動	
BrokerType	ICS、WMQI、WAS			
CharacterEncoding	ascii7、ascii8、SJIS、Cp949、GBK、Big5、Cp297、Cp273、Cp280、Cp284、Cp037、Cp437 **注:** これは、サポートされる値の一部です。	ascii7	コンポーネント再始動	
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント再始動	Repository Directory は <REMOTE>
ContainerManagedEvents	値なしまたは JMS	値なし	コンポーネント再始動	Delivery Transport は JMS
ControllerStoreAndForwardMode	true または false	True	動的	Repository Directory は <REMOTE>
ControllerTraceLevel	0 から 5	0	動的	Repository Directory は <REMOTE>
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	コンポーネント再始動	JMS トランスポートのみ
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

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
DuplicateEventElimination	True または False	False	コンポーネント再始動	JMS トランスポートのみ: Container Managed Events は <NONE> でなければならない
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory または CxCommon.Messaging.jms.SonicMQFactory または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	FactoryClassName が IBM の場合は crossworlds.queue.manager を使用。FactoryClassName が Sonicの場合 localhost:2506 を使用。	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMaxNativeStackSize	スタックのサイズ (キロバイト単位)	128k	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	Repository Directory は <REMOTE>
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならない
Locale	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR **注:** これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	True または False	False	コンポーネント再始動	Repository Directory は <REMOTE> でなければならない

表 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
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならない
MessageFileName	パスまたはファイル名	InterchangeSystem.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	*CONNECTORNAME*/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は True でなければならない
OADAutoRestartAgent	True または False	False	動的	Repository Directory は <REMOTE> でなければならない
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければならない
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければならない
PollEndTime	HH:MM	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒) no (ポーリングを使用不可にする) key (コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力された場合にのみポーリングする)	10000	動的	
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

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
RequestQueue	有効な JMS キュー名	*CONNECTORNAME*/REQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
ResponseQueue	有効な JMS キュー名	*CONNECTORNAME*/RESPONSEQUEUE	コンポーネント再始動	Delivery Transport が JMS の場合: Repository Directory が <REMOTE> の場合のみ必要
RestartRetryCount	0 から 99	3	動的	
RestartRetryInterval	適切な正数 (単位: 分) : 1 から 2147483547	1	動的	
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 CwXML。Repository Directory が <REMOTE> であれば CwBO
WsifSynchronousRequest Timeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	WAS のみ
XMLNameSpaceFormat	short、long	short	エージェント再始動	WebSphere MQ Message Brokers および WAS のみ

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

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

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

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

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

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

AgentTraceLevel

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

ApplicationName

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

BrokerType

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

CharacterEncoding

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

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

デフォルトでは、ドロップ・リストには、サポートされる文字エンコードの一部のみが表示されます。ドロップ・リストに、サポートされる他の値を追加するには、

製品ディレクトリーにある ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。詳細については、Connector Configurator に関する付録を参照してください。

ConcurrentEventTriggeredFlows

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

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

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

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

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

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

ContainerManagedEvents

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

デフォルト値は No value です。

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

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

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

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

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

ControllerStoreAndForwardMode

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

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

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

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

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

デフォルト値は true です。

ControllerTraceLevel

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

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

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

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

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

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

WebSphere MQ および IDL

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

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

JMS

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

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

このプロパティーは、false に設定することもできます。

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

FaultQueue

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

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

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

jms.NumConcurrentRequests

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

デフォルト値は 10 です。

jms.Password

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

デフォルトはありません。

jms.UserName

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

デフォルトはありません。

ListenerConcurrency

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

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

Locale

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

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、以下のように説明されます。

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

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

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

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

LogAtInterchangeEnd

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

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

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

MaxEventCapacity

コントローラー・バッファー内のイベントの最大数。このプロパティーはフロー制御が使用し、RepositoryDirectory プロパティーの値が <REMOTE> の場合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 ¥connectors¥messages です。メッセージ・ファイルが標準位置に格納されていない場合は、メッセージ・ファイル名を絶対パスで指定します。

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

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

MonitorQueue

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

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

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

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

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

デフォルト値は false です。

OADMaxNumRetry

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

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

デフォルト値は 1000 です。

OADRetryTimeInterval

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

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

デフォルト値は 10 です。

PollEndTime

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

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

PollFrequency

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

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

デフォルト値は 10000 です。

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

PollQuantity

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

PollStartTime

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

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

RequestQueue

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

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

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

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

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

ResponseQueue

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

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

RestartRetryCount

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

デフォルト値は 3 です。

RestartRetryInterval

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

デフォルト値は 1 です。

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、62 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

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

WsifSynchronousRequest Timeout

WAS 統合ブローカーでのみ使用されます。

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

デフォルト値は 0 です。

XMLNameSpaceFormat

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

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

デフォルト値は short です。

付録 B. Connector Configurator

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

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

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

注:

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

この付録では、次のトピックについて説明します。

Connector Configurator の概要

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

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

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

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成します。
- **コネクター構成ファイル**を作成します。インストールするコネクターごとに構成ファイルを 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します。
  場合によっては、コネクター・テンプレートでプロパティーに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、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

© Copyright IBM Corp. 2004

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、76 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。

ブローカーが IBM WebSphere InterChange Server の場合に Connector Configurator を実行するにはのためには、以下の手順を実行します。

- **「スタート」>「プログラム」**から、**「IBM WebSphere InterChange Server」>「IBM WebSphere Business Integration Toolset」>「開発」>「Connector Configurator」**をクリックします。

- **「ファイル」>「新規」>「構成ファイル」**を選択します。
- **「システム接続: 統合ブローカー」**の隣のプルダウン・メニューをクリックすると、ICS 接続を選択できます。

WebSphere Business Integration Adapters および他のブローカーがインストールされている場合に Connector Configurator を実行するには、以下の手順を実行します。

- **「スタート」>「プログラム」**から、**「IBM WebSphere Business Integration Adapters」>「ツール」>「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: 統合ブローカー」**の隣のプルダウン・メニューをクリックすると、使用しているブローカーに応じて、WMQI 接続または WAS 接続を選択できます。

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 (81 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

1. 「System Manager」ウィンドウの「コネクター」フォルダーで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。
2. 「標準のプロパティー」タブをクリックし、この構成ファイルに含まれているプロパティーを確認します。

コネクター固有のプロパティー・テンプレートの作成

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

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

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

新規テンプレートの作成

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

テンプレートは以下のように作成します。

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

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

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

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

一般特性の指定

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

- **一般:**
  プロパティー・タイプ
  更新されたメソッド
  説明

- **フラグ**
  標準フラグ
- **カスタム・フラグ**
  フラグ

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

値の指定

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

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

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

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

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

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

依存関係の設定

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

依存プロパティーは、別のプロパティーの値が特定の条件に合致する場合にのみ、テンプレートに組み込まれて、構成ファイルで使用されるプロパティーです。例えば、テンプレートに `PollQuantity` が表示されるのは、トランスポート機構が JMS

であり、DuplicateEventElimination が True に設定されている場合のみです。
プロパティーを依存プロパティーとして指定し、依存する条件を設定するには、以下のステップを実行します。

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

   == (等しい)

   != (等しくない)

   > (より大)

   < (より小)

   >= (より大か等しい)

   <= (より小か等しい)
4. **「条件値」**フィールドで、依存プロパティーをテンプレートに組み込むために必要な値を入力します。
5. **「使用可能なプロパティー」**表示で依存プロパティーを強調表示させて矢印をクリックし、**「依存プロパティー」**表示に移動させます。
6. **「完了」**をクリックします。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。

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

ブローカーを選択するには、以下のステップを実行します。

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

また、以下の作業も実行できます。

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

コネクター固有のテンプレートからの構成ファイルの作成

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

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

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

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

     ICS 接続、WebSphere Message Brokers 接続、WAS のいずれかをクリックします。
   - **「コネクター固有プロパティー・テンプレート」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 84 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。

- **「更新メソッド」**フィールドは通知用であり、構成できません。このフィールドは、値が変更されたプロパティーをアクティブにするために必要なアクションを示します。

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下の手順を実行します。

1. 値を設定するフィールド内でクリックします。
2. 値を入力するか、ドロップダウン・メニューが表示された場合にはメニューから値を選択します。
3. 標準のプロパティーの値をすべて入力後、以下のいずれかを実行することができます。
   - 変更内容を破棄し、元の値を保持したままで Connector Configurator 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. 『標準コネクター・プロパティーの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

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

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

コネクター・プロパティーの暗号化

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

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

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

更新メソッド

付録 A『*コネクターの標準構成プロパティー*』の 56 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、以下の手順を実行します。

1. **「ビジネス・オブジェクト名」**リストで空のフィールドをクリックします。System Manager プロジェクトに存在するすべてのビジネス・オブジェクト定義を示すドロップダウン・リストが表示されます。
2. 追加するビジネス・オブジェクトをクリックします。

3. ビジネス・オブジェクトの**「エージェント・サポート」**(以下で説明) を設定します。
4. 「Connector Configurator」ウィンドウの「ファイル」メニューで、**「プロジェクトに保管」**をクリックします。追加したビジネス・オブジェクト定義に指定されたサポートを含む、変更されたコネクター定義が、System Manager のプロジェクトに保管されます。

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

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

System Manager を実行している場合、**「サポートされているビジネス・オブジェクト」**タブの**「ビジネス・オブジェクト名」**列の下にある空のボックスを選択できます。コンボ・ボックスが表示され、コネクターが属する統合コンポーネント・ライ

ブラリー・プロジェクトから選択可能なビジネス・オブジェクトのリストが示されます。リストから必要なビジネス・オブジェクトを選択します。

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

ご使用のブローカーが WAS の場合

使用するブローカー・タイプとして WebSphere Application Server を選択した場合、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。

  以下の手順を実行して、マップを明示的にバインドします。

  1. **「明示的 (Explicit)」**列で、バインドするマップのチェック・ボックスにチェックマークを付けます。
  2. ビジネス・オブジェクトに関連付けるマップを選択します。
  3. 「Connector Configurator」ウィンドウの**「ファイル」**メニューで、**「プロジェクトに保管」**をクリックします。
  4. プロジェクトを ICS に配置します。
  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、以下の手順を実行します。

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

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

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

データ・ハンドラー

データ・ハンドラー・セクションの構成が使用可能となるのは、DeliveryTransport の値に JMS を、また ContainerManagedEvents の値に JMS を指定した場合のみです。すべてのアダプターでデータ・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。

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

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

構成ファイルの変更

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

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

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

- Connector Configurator で既存の構成ファイルを開きます。
- **「標準のプロパティー」**タブを選択します。
- 「標準のプロパティー」タブの**「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>
```

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。

© Copyright IBM Corp. 2004

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

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

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

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

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

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

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

プログラミング・インターフェース情報

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

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

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

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

商標

以下は、IBM Corporation の商標です。

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

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

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

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

IBM WebSphere Business Integration Adapter Framework V2.4.0