WebSphere Business Integration Adapters

Adapter for FIX Protocol ユーザーズ・ガイド

V1.7.0

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、 105 ページの『付録 C. 特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere Business Integration Adapter for FIX Protocol バージョン 1.7.0、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： WebSphere Business Integration Adapters
Adapter for FIX Protocol User Guide
V1.7.0

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1刷 2004.1

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

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

本書について

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

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

対象読者

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

本書の前提条件

本書を利用するには、WebSphere Business Integration システム、ビジネス・オブジェクト開発、MQSeries アプリケーション、および FIX Protocol について十分な知識を持っていることが必要です。

関連資料

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。

本書には、次の資料「*システム・インストール・ガイド (Windows 版)*」、「*システム・インストール・ガイド (UNIX 版)*」、および「*WebSphere InterChange Server インプリメンテーション・ガイド*」への参照が多数含まれています。本書を印刷する場合には、これらの資料も印刷することをお勧めします。

以下のサイトから、関連資料をインストールすることができます。

- 一般的なアダプター情報が必要な場合、アダプターを WebSphere Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) とともに使用する場合、およびアダプターを WebSphere Application Server とともに使用する場合は、以下のサイトを参照してください。

  http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- アダプターを InterChange Server とともに使用する場合は、以下のサイトを参照してください。

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

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

- Message Broker (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) の詳細については、以下のサイトを参照してください。
  http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- WebSphere Application Server の詳細については、以下を参照してください。
  http://www.ibm.com/software/webservers/appserv/library.html

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

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 font`	コマンド名、ファイル名、入力情報、システムが画面に出力した情報など、記述されたとおりの値を示します。
太字	初出語を示します。
イタリック、*イタリック*	変数名または相互参照を示します。
青い文字	オンラインで表示したときにのみ見られる青の部分は、相互参照用のハイパーリンクです。青いテキストをクリックすることにより、参照先オブジェクトに飛ぶことができます。
{ }	構文の記述行の場合、中括弧 {} で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
[ ]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプションのパラメーターです。
...	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、`option[,...]` は、複数のオプションをコンマで区切って指定できることを意味します。
< >	命名規則では、不等号括弧は名前の個々の要素を囲み、各要素を区別します。(例: `<server_name><`*`connector_name`*`>tmp.log`)
/、¥	本書では、ディレクトリー・パスに円記号 (¥) を使用します。UNIX システムの場合には、円記号をスラッシュ (/) に置き換えてください。すべての IBM CrossWorlds 製品パス名は、製品がシステムにインストールされているディレクトリーを基準にした相対パス名です。
UNIX: および Windows:	これらのいずれかで始まる段落は、オペレーティング・システム間の相違を列挙した注記です。
u	この記号は、**UNIX/Windows** の段落の終わりを示します。また、複数段落にわたる注釈の終わりを示します。
%text% および *$text*	% 記号で囲まれたテキストは、Windows の `text` システム変数またはユーザー変数の値を示します。UNIX 環境での同等の表記は *$text* です。これは、UNIX 環境変数 *text* の値を示します。
ProductDir	製品のインストール先ディレクトリーを表します。

本リリースの新機能

リリース 1.7.0 の新機能

アダプターで、FIX Protocol バージョン 4.4 がサポートされました。

コネクターは、以下のプラットフォーム上で実行されます。

- Microsoft Windows 2000
- Solaris 7、8 または AIX 5.1、5.2 または HP UX 11.i

バージョン 1.7.0 以降の Adapter for FIX Protocol は Microsoft Windows NT ではサポートされなくなりました。

アダプターのインストール情報は、本書から移動しました。この情報の新たな入手先については、第 2 章を参照してください。

リリース 1.6.x の新機能

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

コネクターは、以下のプラットフォーム上で実行されます。

- Microsoft Windows NT 4.0 Service Pack 6A または Windows 2000
- Solaris 7、8 または AIX 5.1、5.2 または HP UX 11.i

リリース 1.5.x の新機能

2003 年 3 月更新。「CrossWorlds」という名前を使用してシステム全体を示したりコンポーネントまたはツールの名前を変更したりすることはなくなりましたが、その他の点では以前とほとんど同じです。例えば、「CrossWorlds System Manager」は現在は「System Manager」で、「CrossWorlds InterChange Server」は現在は「WebSphere InterChange Server」です。

データ・ハンドラーを入力キューと関連付けることができるようになりました。詳細については、32 ページの『InputQueue へのデータ・ハンドラーのマッピング』を参照してください。

保証付きイベント・デリバリー機能が拡張されました。詳細については、11 ページの『保証付きイベント・デリバリー』を参照してください。

リリース 1.4.x の新機能

本書では、InterChange Server (ICS) と MQ Integrator のいずれかの WebSphere 統合ブローカーと共に本アダプターを使用するための情報を提供しています。

リリース 1.2.x の新機能

コネクターは国際化されました。詳細については、 15 ページの『ロケール依存データの処理』および 69 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。

リリース 1.1.x の新機能

IBM(R) WebSphere(R) Business Integration Adapter for FIX Protocol には、Connector for FIX Protocol が含まれています。このアダプターは、InterChange Server (ICS) と WebSphere MQ Integrator (WMQI) の両方の統合ブローカーと共に動作します。統合ブローカーとは、異種のアプリケーション・セット間の統合を実行するアプリケーションです。統合ブローカーは、データ・ルーティングなどのサービスを提供します。アダプターには、以下の要素が含まれます。

- FIX Protocol に固有のアプリケーション・コンポーネント
- ビジネス・オブジェクトのサンプル
- IBM WebSphere Adapter フレームワーク。コンポーネントは以下のとおりです。
  - コネクター・フレームワーク
  - 開発ツール (Business Object Designer と Connector Configurator を含む)
  - API (CDK を含む)

本書では、このアダプターを ICS と WMQI の両方の統合ブローカーと共に使用するための情報を提供します。

重要: コネクターは国際化に対応していないため、ISO Latin-1 データのみが処理されることが確実である場合を除いて、コネクターと ICS バージョン 4.1.1 を併用しないでください。

目次

第 1 章 概要

Connector for FIX Protocol は、WebSphere Business Integration Adapter for FIX Protocol のランタイム・コンポーネントです。このコネクターを使用すると、WebSphere 統合ブローカーと FIX 対応のビジネス・プロセスとの間でビジネス・オブジェクトを交換できるようになります。

注: WebSphere では、FIX バージョン 4.0、4.1、4.2、4.3、および 4.4 のメッセージに対応するビジネス・オブジェクトをサポートしています。

コネクターは、アプリケーション固有のコンポーネントとコネクター・フレームワークから成り立っています。アプリケーション固有のコンポーネントには、特定のアプリケーションに応じて調整されたコードが含まれます。コネクター・フレームワークは統合ブローカーとアプリケーション固有のコンポーネントの間の仲介役として機能し、そのコードはどのコネクターにも共通です。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントとの間で以下のようなサービスを提供します。

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

本書では、アプリケーション固有のコンポーネントおよびコネクター・フレームワークについて説明します。本書では、この 2 つのコンポーネントをまとめてコネクターと呼びます。

WebSphere Business Integration Adapter は、いずれも統合ブローカーとともに動作します。Connector for FIX は、InterChange Server (ICS) と WebSphere MQ Integrator (WMQI) の両方の統合ブローカーとともに動作します。

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

注: WebSphere Business Integration Adapter は、いずれも統合ブローカーとともに動作します。Connector FIX Protocol は、以下の統合ブローカーとともに動作します。

- WebSphere InterChange Server (ICS) 統合ブローカー。詳細については、「*テクニカル入門 (IBM WebSphere InterChange Server)*」を参照してください。

- WebSphere MQ Integrator (WMQI) 統合ブローカー。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) 統合ブローカー。詳細は、「アダプター実装ガイド *(WebSphere Application Server)*」を参照してください。

Connector for FIX Protocol を使用することで、ICS、WMQI、WAS のいずれかの統合ブローカーと、FIX メッセージの形式でデータを送受信するアプリケーションとの間で、ビジネス・オブジェクトを交換できるようになります。

Adapter for FIX の環境

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

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。FIX Protocol のアダプターのバージョン 1.7 は、以下のアダプター・フレームワークと統合ブローカーでサポートされています。

- アダプター・フレームワーク: WebSphere Business Integration Adapter Framework の以下のバージョン:
  - 2.1
  - 2.2
  - 2.3.x
  - 2.4
- 統合ブローカー:
  - WebSphere InterChange Server の以下のバージョン:
    - 4.11
    - 4.2
    - 4.2.1
    - 4.2.x
  - WebSphere MQ Integration Broker、バージョン 2.1.0
  - WebSphere Business Integration Message Broker、バージョン 5.0
  - WebSphere Application Server Enterprise、バージョン 5.0.2 (WebSphere Studio Application Developer Integration Edition、バージョン 5.0.1 と併用)

例外については、「リリース情報」を参照してください。

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

WebSphere InterChange Server (ICS) については、「*システム・インストール・ガイド (UNIX 版)*」または「*システム・インストール・ガイド (Windows 版)*」を参照してください。

WebSphere Message Brokers (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) については、「*WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド*」および WebSphere Message Broker のインストールに関する資料を参照してください。一部は次の Web サイト
http://www.ibm.com/software/integration/mqfamily/library/manualsa/ で入手可能です。
WebSphere Application Server については、「アダプター実装ガイド (WebSphere Application Server)」および次の資料を参照してください。
http://www.ibm.com/software/webservers/appserv/library.html

アダプターの規格

アダプターは、以下のバージョンの FIX メッセージ・セットに対応できるように作成されています。

- 4.0
- 4.1
- 4.2
- 4.3
- 4.4

アダプターのプラットフォーム

アダプターは、以下のプラットフォームでサポートされます。

- Windows 2000
- AIX 5.1、5.2
- Solaris 7、8
- HP-UX 11i

アダプターの依存関係

アダプターには、以下のソフトウェア前提条件などの依存関係があります。

- アダプターは、WebSphere MQ Integrator Broker バージョン 2.1 および WebSphere InterChange Server バージョン 4.2 でサポートされています。
- WebSphere Business Integration Message Broker バージョン 5.0 を使用している場合は、CSD02 をこの統合ブローカーにインストールする必要があります。

- コネクターは、WebSphere MQ、または WebSphere MQ 5.1、5.2 [1]および 5.3 を介してアプリケーションとのインターオペラビリティーをサポートします。そのため、次のいずれかのソフトウェア・リリースをインストールしている必要があります。

  注: 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。
- さらに、IBM WebSphere MQ Java クライアント・ライブラリーも必要です。

ロケール依存データ

コネクターは、2 バイト文字セットをサポートして、指定された言語でメッセージ・テキストを送れるように国際化されています。コネクターは、1 つの文字コードを使用する場所から別のコード・セットを使用する場所にデータを転送するとき、データの意味を保存するように文字変換を実行します。

Java 仮想マシン (JVM) 内の Java ランタイム環境は、Unicode 文字コード・セットでデータを表現します。Unicode には、最も広く知られている文字コード・セット (単一バイトおよびマルチバイトの両方) の文字のエンコードが含まれています。WebSphere Business Integration システム内のほとんどのコンポーネントは、Java で書かれています。そのため、統合コンポーネント間でデータを転送する際に、ほとんどの場合文字変換は必要ありません。

エラー・メッセージや通知メッセージを個々の国や地域に合った適切な言語で記録するには、個々の環境に合わせて `Locale` 標準構成プロパティーを構成する必要があります。構成プロパティーの詳細については、69 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

コネクターはメタデータ主導型です。メッセージ・ルーティングおよびフォーマット変換は、イベント・ポーリング手法によって開始されます。コネクターは、MQ にインプリメントされている JavaTM Message Service (JMS) を使用します。JMS はエンタープライズ・メッセージング・システムにアクセスするための API です。

コネクターを使用すると、FIX (Financial Information eXchange) メッセージを送受信するアプリケーションと、統合ブローカーとの間で、データの変更が発生した場合にビジネス・オブジェクトを非同期的に交換できるようになります。

FIX プロトコルは、証券取引情報のリアルタイムでの電子的交換を特に目的として開発された、メッセージング規格です。FIX は、FIX Protocol, Ltd. が所有し保守しているパブリック・ドメインの仕様です。

1. ご使用の環境で、文字セットの変換に取得時変換方式がインプリメントされている場合は、IBM から最新の MA88 (JMS クラス) をダウンロードする必要があります。パッチ・レベルは、5.2.2 以上 (WebSphere MQ バージョン 5.2 の場合) とします。これにより、エンコードがサポートされていないことによるエラーを回避できる場合があります。

コネクターは、キューから FIX メッセージを取得し、FIX データ・ハンドラーを呼び出してそのメッセージを対応するビジネス・オブジェクトに変換し、統合ブローカーにそのビジネス・オブジェクトを送達します。反対方向の場合、コネクターは統合ブローカーからビジネス・オブジェクトを受信し、同じデータ・ハンドラーを使用してそれらを FIX メッセージに変換し、その後メッセージを WebSphere MQ キューに送達します。

メッセージ処理に使用されるビジネス・オブジェクトと動詞のタイプは、FIX メッセージ・ヘッダーに含まれる FORMAT フィールドを基本としています。コネクターはメタオブジェクト項目を使用して、ビジネス・オブジェクト名および動詞を決定します。FIX メッセージ・ヘッダーの FORMAT フィールド・テキストに関連付けるビジネス・オブジェクト名と動詞を保管するために、メタオブジェクトを構成してください。

オプションで、コネクターに渡されるビジネス・オブジェクトに子として追加される動的なメタオブジェクトを構成することもできます。子メタオブジェクトの値は、コネクター全体を対象として指定されている静的なメタオブジェクトに指定されている値をオーバーライドします。子メタオブジェクトが定義されていない場合や、必要な変換プロパティーが定義されていない場合は、コネクターはデフォルトで、その値に対する静的なメタオブジェクトを調べます。単一の静的なコネクター・メタオブジェクトの代わりに、またはそれを補足するために、1 つ以上の動的な子メタオブジェクトを指定することができます。

コネクターは複数の入力キューのポーリングが可能で、各キューをラウンドロビン方式でポーリングして、各キューから指定された数のメッセージを検索します。ポーリング中に検索するメッセージごとに、コネクターは動的な子メタオブジェクト (ビジネス・オブジェクトに指定されている場合) を追加します。子メタオブジェクトの値はコネクターに対し、メッセージ・フォーマットとメッセージが検索された入力キューの名前を、属性に取り込むよう指示できます。

入力キューからメッセージを検索する場合、コネクターは、メッセージ・ヘッダーに含まれる FORMAT フィールドに対応するビジネス・オブジェクト名を探します。次に、メッセージ本文が適切なビジネス・オブジェクトの新しいインスタンスとともにデータ・ハンドラーに渡されます。メッセージ・フォーマットに対応するビジネス・オブジェクト名が見つからない場合、メッセージ本文のみがデータ・ハンドラーに渡されます。例えば、ICS が統合ブローカーとして使用されている場合、コネクターは、ビジネス・オブジェクトへのメッセージ内容の取り込みが正常に完了すると、そのビジネス・オブジェクトがサブスクライブされているかどうかを確認してから、gotApplEvents() メソッドを使用してそのビジネス・オブジェクトを ICS に配信します。

アプリケーションとコネクターの通信方式

コネクターは、IBM WebSphere MQ にインプリメントされている Java Message Service (JMS) 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。

メッセージ要求

図 1 にメッセージ要求通信を示します。doVerbFor() メソッドが統合ブローカーからビジネス・オブジェクトを受信すると、コネクターはビジネス・オブジェクトをデータ・ハンドラーに渡します。データ・ハンドラーはビジネス・オブジェクトを JMS に適したテキストに変換し、コネクターはそれをメッセージとしてキューに発行します。JMS 層は適切な呼び出しを行ってキュー・セッションを開き、メッセージを発送します。

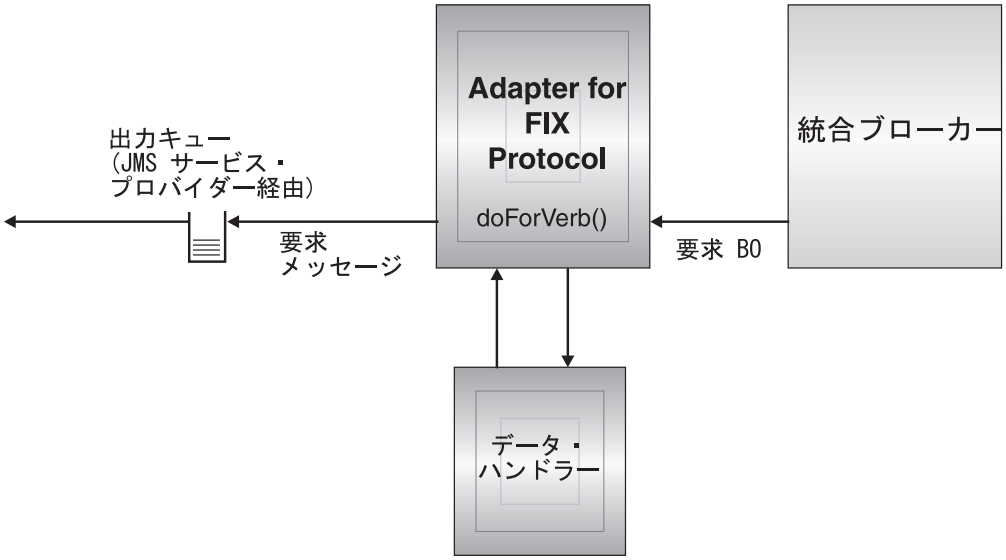

図 1. アプリケーションとコネクターの通信方式: メッセージ要求

メッセージの戻り

図 2 に、メッセージが戻る方向を示します。pollForEvents() メソッドは、次の適用可能なメッセージを入力キューから取得します。メッセージは、進行中キューにステージされ、処理が完了するまでそこにとどまります。コネクターはまず、静的または動的メタオブジェクトを使用してそのメッセージ・タイプがサポートされているかどうかを判断します。メッセージがサポートされている場合、コネクターはメッセージを構成済みのデータ・ハンドラーに渡します。このデータ・ハンドラーは、メッセージをビジネス・オブジェクトに変換します。設定される動詞には、そのメッセージ・タイプに対して設定されている変換プロパティーが反映されます。ICS が統合ブローカーとして使用されている場合、コネクターは、次に、そのビジネス・オブジェクトがいずれかのコラボレーションでサブスクライブされているかどうかを判別します。サブスクライブされている場合、gotApplEvents() メソッドはビジネス・オブジェクトを ICS に配信し、メッセージは進行中のキューから除去されます。

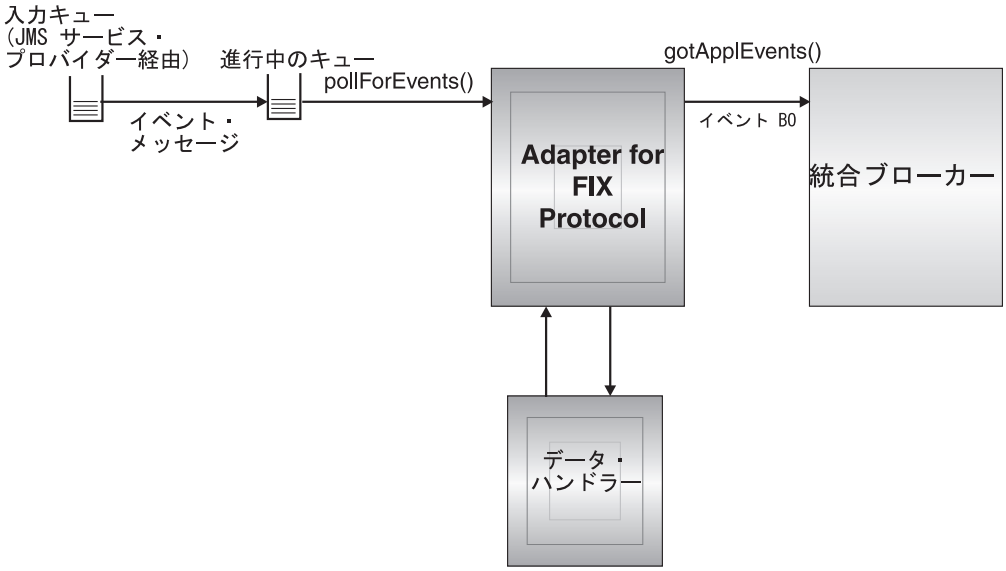

図 2. アプリケーションとコネクターの通信方式: メッセージの戻り

イベント処理

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

検索

コネクターは `pollForEvents()` メソッドを使用して、一定の間隔で MQ キューをポーリングしてメッセージの有無を確認します。メッセージが見つかった場合、コネクターはメッセージを MQ キューから取り出して検討し、そのフォーマットを判断します。メッセージ・フォーマットがコネクターの静的オブジェクト内に定義されている場合、コネクターは、メッセージ本文とそのフォーマットに関連付けられたビジネス・オブジェクトの新しいインスタンスを構成済みのデータ・ハンドラーに渡します。この場合、データ・ハンドラーがビジネス・オブジェクトへのデータの取り込みと動詞の指定を行う必要があります。静的なメタオブジェクト内にメッセージ・フォーマットが定義されていない場合、コネクターはメッセージ本文のみをデータ・ハンドラーに渡します。この場合、データ・ハンドラーは、そのメッセージに応じた適切なビジネス・オブジェクトを決定および作成し、そこにデータを取り込む必要があります。イベント失敗のシナリオについては、55 ページの『エラー処理』を参照してください。

コネクターはメッセージを処理する際に、まず入力キューに対してトランザクションのセッションを開きます。このトランザクションのアプローチによって、ビジネス・オブジェクトが統合ブローカーに 2 回配信される可能性があります。これは、コネクターがビジネス・オブジェクトのサブミットには成功するが、キュー内のトランザクションのコミットに失敗することが原因です。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はキューからイベントが除去される前にすべて実行され、コミットされる必要があります。したがって、コネクターは、キューのメッセージを検索するとき、1) メッセージからビジネス・オブジェクトへの変換、2) 統合ブローカーへのビジネス・オブジェクトの配信、3) 戻り値の受信、の 3 つの処理が完了するまで、検索をコミットしません。

同期イベント処理

Connector for FIX Protocol は、要求を処理した後、その結果の詳細を示すレポート・メッセージを発行してアプリケーションに返送することもできます。これは、WebSphere MQ を通じて発行した要求に対するフィードバックを必要とするアプリケーションをサポートするためです。

この処理を実現するために、コネクターはこのような要求のビジネス・データを統合ブローカーに同期的に通知します。ビジネス・オブジェクトが正常に処理された場合、コネクターは、統合ブローカーからの戻りコードとビジネス・オブジェクトの変更の内容を含むレポートを、要求発行者に返送します。ビジネス・オブジェクトが処理されなかった場合、コネクターは、該当するエラー・コードとエラー・メッセージを含むレポートを送信します。

いずれの場合も、Connector for FIX Protocol に要求を送信するアプリケーションは、要求の結果について通知されます。

処理: Connector for FIX Protocol は、肯定通知レポートまたは否定通知レポート (PAN または NAN) を要求するメッセージを受信すると、そのメッセージの内容を統合ブローカーに同期的に通知します。その後で、戻りコードと変更されたビジネス・データをレポート・メッセージに組み込み、要求元のアプリケーションに返送します。

表 1 に、コネクターで同期的に受信し処理する FIX メッセージに必要な構造を示します。

表 1. 同期 WebSphere MQ メッセージに必要な構造

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

表 1. 同期 WebSphere MQ メッセージに必要な構造 (続き)

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

ICS ベースのコネクターは、 8 ページの表 1 で説明したメッセージを受信すると、以下の処理を行います。

1. 構成されているデータ・ハンドラーを使用して、メッセージの本文に含まれるビジネス・オブジェクトを再構成します。
2. コラボレーション名に指定されたビジネス・プロセスおよび静的メタデータ・オブジェクトの動詞を検索します。
3. 指定されたコラボレーションに、ビジネス・オブジェクトを同期的に通知します。
4. 処理の結果とビジネス・オブジェクトのすべての変更またはエラー・メッセージをカプセル化したレポートを生成します。
5. 要求の replyToQueue および replyToQueueManager フィールド内で指定されたキューに、レポートを送信します。

10 ページの表 2 に、コネクターから要求発行者に返送されるレポートの構造を示します。

表 2. 要求発行者に戻されるレポートの構造

MQMD フィールド	説明	サポートされる値 (複数の値がある場合は OR として扱います)
MessageType	メッセージ・タイプ	REPORT
feedback	レポートのタイプ	次のいずれかです。 • MQRO_PAN ビジネス・オブジェクトが正常に処理された場合に、レポートが返送されます。 • MQRO_NAN要求の処理中にコネクターまたは統合ブローカーがエラーを検出した場合に、レポートが返送されます。
Message Body		ビジネス・オブジェクトが正常に処理された場合、コネクターはメッセージの本文に統合ブローカーから戻されたビジネス・オブジェクトを取り込みます。静的メタデータ・オブジェクト内の DoNotReportBusObj プロパティーを true に設定すると、このデフォルトの振る舞いをオーバーライドできます。要求を処理できなかった場合、コネクターはメッセージの本体にコネクターまたは統合ブローカーによって生成されたエラー・メッセージを取り込みます。

リカバリー

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

fail on startup

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

reprocess

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

ignore

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

log error

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

アーカイブ

コネクター・プロパティー ArchiveQueue が指定されて有効なキューを示す場合、コネクターは正常に処理されたすべてのメッセージのコピーをアーカイブ・キューに入れます。ArchiveQueue が定義されていない場合は、メッセージは処理後に廃棄されます。アンサブスクライブされたメッセージまたはエラーがあるメッセージのアーカイブの詳細については、55 ページの『エラー処理』を参照してください。

注: JMS の規則によると、検索されたメッセージをすぐに別のキューに発行することはできません。メッセージのアーカイブおよび再デリバリーを可能にするためには、コネクターはまず、元のメッセージの本文とヘッダー (該当する場合) を複製した新規のメッセージを作成します。JMS サービス・プロバイダーとの矛盾を避けるために、JMS 必須フィールドのみを複製します。この結果、フォーマット・フィールドが、アーカイブまたは再デリバリーされるメッセージ用に複製された唯一の追加メッセージ・プロパティーです。

保証付きイベント・デリバリー

保証付きイベント・デリバリー機能により、コネクター・フレームワークは、イベントが逸失したり、コネクターのイベント・ストア、JMS イベント・ストア、および宛先の JMS キューの間でイベントが 2 度送信されたりするのを防ぐことができます。JMS を有効にするには、コネクターの DeliveryTransport 標準プロパティーを JMS に設定する必要があります。この構成により、コネクターは JMS トランスポートを使用することになり、コネクターと統合ブローカーの間の後続の通信はすべて、このトランスポートを介して行われます。JMS トランスポートでは、最終的にメッセージが宛先に移送されることが保証されます。JMS トランスポートの役割は、トランザクションのキュー・セッションの開始後、コミットが発行されるまでメッセージが確実にキャッシュされるようにすることです。障害が起こったとき、あるいはロールバックが発行されたときには、メッセージは廃棄されます。

注: 保証付きイベント・デリバリー機能を使用しない場合、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・デリバリー機能は、JMS イベント・ストアを持つ JMS 対応コネクター用、あるいは JMS イベント・ストアを持たない JMS 対応コネクター用に構成できます。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、「*コネクター開発ガイド (Java 用)*」の説明を参照してください。

コネクター・フレームワークがビジネス・オブジェクトを ICS 統合ブローカーに送信できない場合、オブジェクトは UnsubscribedQueue や ErrorQueue ではなく FaultQueue に置かれ、状況表示と問題の説明が生成されます。FaultQueue メッセージは、MQRFH2 形式で書き込まれます。

ビジネス・オブジェクト要求

ビジネス・オブジェクト要求は、統合ブローカーがビジネス・オブジェクトをコネクター・フレームワークに送信する際に処理されます。構成済みのデータ・ハンドラーを使用して、コネクターはビジネス・オブジェクトを FIX メッセージに変換し、それを発行します。データ・ハンドラーの要件を除いては、処理されるビジネス・オブジェクトのタイプに関する要件はありません。

動詞の処理

コネクターは、各ビジネス・オブジェクトの動詞に基づいた統合ブローカーによって、渡されるビジネス・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・オブジェクト動詞をサポートします。

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

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

Create、Update、および Delete

Create、Update、および Delete の各動詞を含むビジネス・オブジェクトの処理は、オブジェクトが非同期または同期のどちらの方式で発行されているかによって異なります。

非同期デリバリー

Create、Update、および Delete の各動詞を含むビジネス・オブジェクトに関しては、これがデフォルトのデリバリー・モードです。メッセージは、データ・ハンドラーを使用してビジネス・オブジェクトから作成され、出力キューに書き込まれます。コネクターは、メッセージが配信されると、BON_SUCCESS、配信されなければ BON_FAIL を戻します。

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

同期デリバリー

コネクター・プロパティーに replyToQueue が定義されていて、ビジネス・オブジェクトの変換プロパティーに responseTimeout が存在する場合、コネクターは同期モードで要求を発行します。コネクターはその後で応答を待機して、受信アプリケーションによって適切なアクションが実行されることを確認します。

FIX の場合、コネクターは最初に、表 3 に示すヘッダーを持つメッセージを発行します。

表 3. 要求メッセージの記述子ヘッダー (MQMD)

フィールド	説明	値
Format	フォーマット名	変換プロパティーで定義された出力フォーマット。IBM 要件に合わせて 8 文字までに切り捨てられます (例: MQSTR)。
MessageType	メッセージ・タイプ	MQMT_DATAGRAM*
Report	必要なレポート・メッセージのオプション	応答メッセージの返送が予測される場合、このフィールドには次の値が取り込まれます。処理が成功したときに肯定処理レポートが必要な場合は、MQRO_PAN*。処理が失敗したときに否定処理レポートが必要な場合は、MQRO_NAN*。生成されるレポートの相関 ID が最初に発行された要求のメッセージ ID と同じになる必要がある場合は、MQRO_COPY_MSG_ID_TO_CORREL_ID*。
ReplyToQueue	応答キューの名前	応答メッセージが期待される場合は、このフィールドにはコネクター・プロパティー ReplyToQueue の値が取り込まれます。
Persistence	メッセージのパーシスタンス (永続性)	MQPER_PERSISTENT*
Expiry	メッセージの存続時間	MQEI_UNLIMITED*

* は、IBM によって定義されている定数を示します。

表 3 に示したメッセージ・ヘッダーの後ろに、メッセージ本文が続きます。メッセージ本文は、データ・ハンドラーを使用して直列化されたビジネス・オブジェクトです。

Report フィールドは、受信側アプリケーションからの肯定処理レポートと否定処理レポート両方の返送が期待されていることを示すよう、設定されます。メッセージを発行したスレッドは、受信アプリケーションが要求を処理できるかどうかを示す応答メッセージを待機します。

アプリケーションは、コネクターから同期要求を受信すると、ビジネス・オブジェクトを処理して、表 4、表 5、および表 6 に示すレポート・メッセージを発行します。

表 4. 応答メッセージの記述子ヘッダー (MQMD)

フィールド	説明	値
Format	フォーマット名	変換プロパティーに定義された busObj の入力フォーマット。
MessageType	メッセージ・タイプ	MQMT_REPORT*
* は、IBM によって定義されている定数を示します。		

表 5. 応答メッセージに取り込まれる内容

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

表 6. FIX のフィードバック・コードと IBM WebSphere Interchange Server の応答値

WebSphere MQ フィードバック・コード	IBM WebSphere Interchange Server の同等な応答*
MQFB_PAN または MQFB_APPL_FIRST	SUCCESS
MQFB_NAN または MQFB_APPL_FIRST + 1	FAIL
MQFB_APPL_FIRST + 2	VALCHANGE
MQFB_APPL_FIRST + 3	VALDUPES
MQFB_APPL_FIRST + 4	MULTIPLE_HITS
MQFB_APPL_FIRST + 5	FAIL_RETRIEVE_BY_CONTENT
MQFB_APPL_FIRST + 6	BO_DOES_NOT_EXIST
MQFB_APPL_FIRST + 7	UNABLE_TO_LOGIN
MQFB_APPL_FIRST + 8	APP_RESPONSE_TIMEOUT (コネクターが即時終了します)
*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。	

ビジネス・オブジェクトが処理できた場合、アプリケーションはフィードバック・フィールドを MQFB_PAN (または特定の IBM CrossWorlds の値) に設定して、レポート・メッセージを作成します。オプションで、アプリケーションはメッセージ本体に変更箇所を含むビジネス・オブジェクトを直列化して取り込むことができます。ビジネス・オブジェクトが処理されなかった場合、アプリケーションはフィードバック・フィールドを MQFB_NAN (または特定の IBM CrossWorlds の値) に設定して、レポート・メッセージを作成します。その後、オプションでメッセージ本体にエラー・メッセージを組み込みます。どちらの場合でも、アプリケーションはメッセージの correlationID フィールドをコネクター・メッセージの messageID に設定し、replyTo フィールドで指定されたキューにメッセージを発行します。

コネクターは、応答メッセージを検索する際に応答メッセージの correlationID と要求メッセージの messageID とを付き合わせます。次に、コネクターは要求を発行したスレッドを通知します。応答のフィードバック・フィールドによって、コネクターはメッセージ本体に、ビジネス・オブジェクトかエラー・メッセージのいずれかが含まれていることを予測します。ビジネス・オブジェクトが要求されたがメッセージ本文に値が含まれていない場合、コネクターは単に、Request 操作で最初に統合ブローカーによって発行されたものと同じビジネス・オブジェクトを戻します。エラー・メッセージが要求され、メッセージ本文に値が含まれていない場合、汎用エラー・メッセージが応答コードとともに統合ブローカーに戻されます。

Retrieve、exists、および retrieve by content

Retrieve、Exists、および Retrieve By Content 動詞を含むビジネス・オブジェクトは、同期デリバリーのみをサポートします。コネクターはこれらの動詞を含むビジネス・オブジェクトを、create、update、および delete 用に定義された同期デリバリ

ーの場合と同様に処理します。ただし、Retrieve、Exists、および Retrieve By Content 動詞を使用している場合、responseTimeout と replyToQueue が必要になります。さらに、Retrieve By Content および Retrieve 動詞の場合は、トランザクションを完了するには、メッセージ本文に直列化ビジネス・オブジェクトが取り込まれる必要があります。

表 7 に、これらの動詞の応答メッセージを示します。

表 7. 応答メッセージに取り込まれる内容

動詞	フィードバック・フィールド	メッセージ本文
Retrieve または RetrieveByContent	FAIL FAIL_RETRIEVE_BY_CONTENT	(オプション) エラー・メッセージ
	MULTIPLE_HITS SUCCESS	直列化ビジネス・オブジェクト
Exist	FAIL	(オプション) エラー・メッセージ
	SUCCESS	

ロケール依存データの処理

コネクターは、2 バイト文字セットをサポートして、指定された言語でメッセージ・テキストを送れるように国際化されています。コネクターは、1 つの文字コードを使用する場所から別のコード・セットを使用する場所にデータを転送するとき、データの意味を保存するように文字変換を実行します。

Java 仮想マシン (JVM) 内の Java ランタイム環境は、Unicode 文字コード・セットでデータを表現します。Unicode には、最も広く知られている文字コード・セット (単一バイトおよびマルチバイトの両方) の文字のエンコードが含まれています。WebSphere Business Integration システム内のほとんどのコンポーネントは、Java で書かれています。そのため、統合コンポーネント間でデータを転送する際に、ほとんどの場合文字変換は必要ありません。

エラー・メッセージや通知メッセージを個々の国や地域に合った適切な言語で記録するには、個々の環境に合わせて Locale 標準構成プロパティーを構成する必要があります。構成プロパティーの詳細については、69 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。

第 2 章 コネクターのインストール、構成、および始動

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

インストール作業の概要

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

統合ブローカーのインストール

この作業には、統合ブローカー・システム (IBM WebSphere InterChange Server (ICS)、WebSphere MQ Integrator (WMQI)、または WebSphere Application Server (WAS)) をインストールする作業が含まれます。コネクター・コンポーネントのインストールの詳細については、ご使用の統合ブローカーに応じて、以下のいずれかのガイドを参照してください。

- 「*システム・インストール・ガイド*」 (ICS を統合ブローカーとして使用する場合)
- 「*WebSphere MQ Integrator Broker 用インプリメンテーション・ガイド*」 (WebSphere MQ Integrator を統合ブローカーとして使用する場合)
- 「*アダプター実装ガイド (WebSphere Application Server)*」(WAS を統合ブローカーとして使用する場合)

Connector for FIX Protocol および関連ファイルのインストール

この作業の内容は、WebSphere ソフトウェア・パッケージからコネクターのファイルを使用システムにインストールすることです。 18 ページの『コネクターおよび関連ファイルのインストール』を参照してください。

コネクターおよび関連ファイルのインストール

WebSphere Business Integration Adapter 製品のインストールについては、「*WebSphere Business Integration Adapters インストール・ガイド*」を参照してください。この資料は、次の Web サイトの WebSphere Business Integration Adapters Infocenter にあります。

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

インストール済みファイルの構造

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

Windows コネクターのファイル構造

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

このユーティリティーは、コネクターを *ProductDir*¥connectors¥FIX ディレクトリーにインストールして、「スタート」メニューにコネクターへのショートカットを追加します。

表 8 に、コネクターが使用する Windows ファイル構造が記載されており、インストーラーを介したコネクターのインストールを選択した際に自動的にインストールされるファイルを示します。

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

***ProductDir* のサブディレクトリー**	**説明**
connectors¥FIX¥CWFIX.jar	FIX Connector
connectors¥FIX¥start_FIX.bat	コネクターの始動ファイル
connectors¥messages¥FIXConnector.txt	コネクター・メッセージ
repository¥FIX¥CN_FIX.txt	コネクター定義
connectors¥FIX¥dependencies¥bonames.cfg	ビジネス・オブジェクト名の決定に使用されるファイル
connectors¥FIX¥dependencies¥FIX40_MessageTypes.cfg	FIX バージョン 4.0 のメッセージ・タイプがすべて含まれている構成ファイル
connectors¥FIX¥dependencies¥FIX41_MessageTypes.cfg	FIX バージョン 4.1 のメッセージ・タイプがすべて含まれている構成ファイル
connectors¥FIX¥dependencies¥FIX42_MessageTypes.cfg	FIX バージョン 4.2 のメッセージ・タイプがすべて含まれている構成ファイル
connectors¥FIX¥dependencies¥FIX43_MessageTypes.cfg	FIX バージョン 4.3 のメッセージ・タイプがすべて含まれている構成ファイル
connectors¥FIX¥dependencies¥FIX40_Tags.cfg	FIX バージョン 4.0 の有効な FIX タグがすべて含まれている構成ファイル
connectors¥FIX¥dependencies¥FIX41_Tags.cfg	FIX バージョン 4.1 の有効な FIX タグがすべて含まれている構成ファイル
connectors¥FIX¥dependencies¥FIX42_Tags.cfg	FIX バージョン 4.2 の有効な FIX タグがすべて含まれている構成ファイル
connectors¥FIX¥dependencies¥FIX43_Tags.cfg	FIX バージョン 4.3 の有効な FIX タグがすべて含まれている構成ファイル

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

***ProductDir* のサブディレクトリー**	**説明**
repository¥FIX¥MO_DataHandler_FIX.txt	FIX データ・ハンドラーに必要なメタオブジェクトが含まれているファイル
repository¥FIX¥MO_DataHandler_Default.txt	FIX データ・ハンドラー用のトップレベル・メタオブジェクト
connectors¥FIX¥samples¥BO_FIX40_ErrorReport.txt	FIX バージョン 4.0 用エラー・レポート・ビジネス・オブジェクトの名前が含まれるファイル
connectors¥FIX¥samples¥BO_FIX41_ErrorReport.txt	FIX バージョン 4.1 用エラー・レポート・ビジネス・オブジェクトの名前が含まれるファイル
connectors¥FIX¥samples¥BO_FIX42_ErrorReport.txt	FIX バージョン 4.2 用エラー・レポート・ビジネス・オブジェクトの名前が含まれるファイル
connectors¥FIX¥samples¥BO_FIX43_ErrorReport.txt	FIX バージョン 4.3 用エラー・レポート・ビジネス・オブジェクトの名前が含まれるファイル
connectors¥FIX¥samples¥BODefinitions¥FIX_4*	ビジネス・オブジェクト定義

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

UNIX コネクターのファイル構造

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

このユーティリティーは、コネクターを *ProductDir*/connectors/FIX ディレクトリーにインストールします。

表 9 には、コネクターが使用する UNIX ファイルの構造が説明されており、インストーラーによるコネクターのインストールを選択したときに自動的にインストールされるファイルが示されています。

表 9. コネクター用としてインストールされた UNIX ファイル構造

***ProductDir* のサブディレクトリー**	**説明**
connectors/FIX/CWFIX.jar	FIX Connector

表 9. コネクター用としてインストールされた UNIX ファイル構造 (続き)

ProductDir のサブディレクトリー	説明
connectors/FIX/start_FIX.sh	コネクターの始動スクリプト。このスクリプトは、汎用のコネクター・マネージャー・スクリプトから呼び出されます。Connector Configurator (WMQI を統合ブローカーとして使用している場合) または CSM の「コネクター構成」画面 (ICS を統合ブローカーとして使用している場合) で「インストール (Install)」をクリックすると、このコネクター・マネージャー・スクリプト用にカスタマイズされたラッパーが、インストーラーによって作成されます。コネクターが ICS と連動する場合、このカスタマイズされたラッパーはコネクターの始動と停止にのみ使用します。コネクターが WMQI と連動する場合は、このカスタマイズされたラッパーはコネクターの始動にのみ使用し、コネクターの停止には mqsiremotestopadapter を使用します。
connectors/messages/FIXConnector.txt	コネクター・メッセージ
repository/FIX/CN_FIX.txt	コネクター定義
connectors/FIX/dependencies/bonames.cfg	ビジネス・オブジェクト名の決定に使用されるファイル
connectors/FIX/dependencies/FIX40_MessageTypes.cfg	FIX バージョン 4.0 のメッセージ・タイプがすべて含まれている構成ファイル
connectors/FIX/dependencies/FIX41_MessageTypes.cfg	FIX バージョン 4.1 のメッセージ・タイプがすべて含まれている構成ファイル
connectors/FIX/dependencies/FIX42_MessageTypes.cfg	FIX バージョン 4.2 のメッセージ・タイプがすべて含まれている構成ファイル
connectors/FIX/dependencies/FIX43_MessageTypes.cfg	FIX バージョン 4.3 のメッセージ・タイプがすべて含まれている構成ファイル
connectors/FIX/dependencies/FIX40_Tags.cfg	FIX バージョン 4.0 の有効な FIX タグがすべて含まれている構成ファイル
connectors/FIX/dependencies/FIX41_Tags.cfg	FIX バージョン 4.1 の有効な FIX タグがすべて含まれている構成ファイル
connectors/FIX/dependencies/FIX42_Tags.cfg	FIX バージョン 4.2 の有効な FIX タグがすべて含まれている構成ファイル
connectors/FIX/dependencies/FIX43_Tags.cfg	FIX バージョン 4.3 の有効な FIX タグがすべて含まれている構成ファイル
repository/FIX/MO_DataHandler_FIX.txt	FIX データ・ハンドラーに必要なメタオブジェクトが含まれているファイル
repository/FIX/MO_DataHandler_Default.txt	FIX データ・ハンドラー用のトップレベル・メタオブジェクト
connectors/FIX/samples/BO_FIX40_ErrorReport.txt	FIX バージョン 4.0 用エラー・レポート・ビジネス・オブジェクトの名前が含まれるファイル
connectors/FIX/samples/BO_FIX41_ErrorReport.txt	FIX バージョン 4.1 用エラー・レポート・ビジネス・オブジェクトの名前が含まれるファイル

表 9. コネクター用としてインストールされた UNIX ファイル構造 (続き)

***ProductDir* のサブディレクトリー**	**説明**
connectors/FIX/samples/BO_FIX42_ErrorReport.txt	FIX バージョン 4.2 用エラー・レポート・ビジネス・オブジェクトの名前が含まれるファイル
connectors/FIX/samples/BO_FIX43_ErrorReport.txt	FIX バージョン 4.3 用エラー・レポート・ビジネス・オブジェクトの名前が含まれるファイル
connectors/FIX/samples/BODefinitions/FIX_4*	ビジネス・オブジェクト定義

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

コネクターの構成

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

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

- Connector Configurator の説明と手順については、87 ページの『付録 B. Connector Configurator』を参照してください。
- 標準コネクター・プロパティーについては、『標準コネクター・プロパティー』と 69 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。
- コネクター固有プロパティーについては、22 ページの『コネクター固有のプロパティー』を参照してください。

コネクターは、始動時に構成値を取得します。実行時セッション中に、1 つまたは複数のコネクター・プロパティーの値を変更することがあります。AgentTraceLevel などの一部のコネクター構成プロパティーの変更は、変更内容が即時に有効になります。他のコネクター・プロパティーの変更は、変更後にコンポーネントまたはシステムを再始動する必要があります。あるプロパティーが動的 (即時に有効になる) か静的 (コネクター・コンポーネントまたはシステムを再始動する必要がある) かを判別するには、Connector Configurator の「コネクター・プロパティー」ウィンドウ内の「更新メソッド」列を参照してください。

標準コネクター・プロパティー

標準構成プロパティーにより、すべてのコネクターによって使用される情報が提供されます。これらのプロパティーの資料については 69 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

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

コネクター固有のプロパティー

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

表 10 に、コネクターに対するコネクター固有の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。

表 10. コネクター固有の構成プロパティー

名前	指定可能な値	デフォルト値	必須
ApplicationPassword	*ログイン・パスワード*		なし
ApplicationUserName	*ログイン・ユーザー ID*		なし
ArchiveQueue	*正常に処理されたメッセージのコピーが送信されるキュー*	`queue://crossworlds.queuemanager/MQCONN.ARCHIVE`	なし
CCSID	*キュー・マネージャーの接続に使用する文字セット*	`null`	なし
Channel	*MQ サーバー・コネクター・チャネル*		はい
ConfigurationMetaObject	*静的な構成メタオブジェクトの名前*		はい
DataHandlerClassName	*データ・ハンドラー・クラス名*	`com.crossworlds.DataHandlers.fix.FixDataHandler`	なし
DataHandlerConfigMO	*データ・ハンドラー・メタオブジェクト*	`MO_DataHandler_Default_FIX`	はい
DataHandlerMimeType	*ファイルの MIME タイプ*	`fix`	なし
ErrorQueue	*未処理のメッセージのキュー*	`queue://crossworlds.queuemanager/MQCONN.ERROR`	なし
HostName	*WebSphere MQ サーバー*		はい

InDoubtEvents	`FailOnStartup`
	`Reprocess`
	`Ignore`
	`LogError`
		`Reprocess`	なし

InputQueue	*ポーリング・キュー*	`queue://crossworlds.queuemanager/MQCONN.IN`	なし
InProgressQueue	*進行中イベント・キュー*	`queue://crossworlds.queuemanager/MQCONN.IN_PROGRESS`	はい
PollQuantity	*InputQueue プロパティーで指定された各キューから取り出すメッセージの数*	`1`	なし
Port	*WebSphere MQ リスナーのために確立するポート*		はい
ReplyToQueue	*コネクターからの要求発行時に応答メッセージが配信されるキュー*	`queue://crossworlds.queuemanager/MQCONN.REPLYTO`	なし
UnsubscribedQueue	*アンサブスクライブされたメッセージが送信されるキュー*	`queue://crossworlds.queuemanager/MQCONN.UNSUBSCRIBE`	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。

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。*

ArchiveQueue
正常に処理されたメッセージのコピーが送信されるキューです。

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

CCSID
キュー・マネージャーの接続に使用する文字セット。このプロパティーの値は、キュー URI の CCSID プロパティーの値と一致する必要があります。 27 ページの『キューの Uniform Resource Identifiers (URI)』を参照してください。

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 がブランクのままか、除去された場合は、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。*

ConfigurationMetaObject
コネクターの構成情報を含む静的なメタオブジェクトの名前です。静的および動的メタオブジェクトの詳細については、29 ページの『メタオブジェクト属性構成』を参照してください。

デフォルト = なし。

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

デフォルト = com.crossworlds.DataHandlers.fix.FixDataHandler

DataHandlerConfigMO
構成情報を提供するためにデータ・ハンドラーに渡されるメタオブジェクトです。

デフォルト = MO_DataHandler_Default

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

デフォルト = fix

ErrorQueue
処理されなかったメッセージが送信されるキューです。

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

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

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

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

pollQuantity の値は 2 のため、コネクターは pollForEvents への呼び出しを行うたびに各キューから多くても 2 つのメッセージを検索します。最初のサイクル (2 回のうちの 1 回目) では、コネクターは、MyQueueA、MyQueueB、および MyQueueC の各キューの 1 番目のメッセージを検索します。これで 1 回目のポーリングは完了し、pollQuantity の値を 1 に設定している場合、コネクターは停止します。ここでは pollQuantity を 2 に設定しているため、コネクターは 2 回目のポーリング (2 回のうちの 2 回目) を開始し、MyQueueA と MyQueueC からそれぞれ 1 つずつメッセージを検索します。MqQueueB は空になっているのでスキップされます。すべてのキューを 2 回ずつポーリングしたら、メソッド pollForEvents への呼び出しは完了します。以下に、メッセージを取り出す順序を示します。

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

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

InProgressQueue
処理中にメッセージを保持するメッセージ・キュー。

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

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

デフォルト = 1

Port
WebSphere MQ リスナーのために確立するポート。

デフォルト = なし。

ReplyToQueue
コネクターからの要求発行時に応答メッセージが配信されるキューです。

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

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 = queue://crossworlds.queue.manager/MQCONN.UNSUBSCRIBED

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

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

複数のコネクター・インスタンスの作成

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

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

新規ディレクトリーの作成

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

```
ProductDir¥connectors¥connectorInstance
```

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

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

```
ProductDir¥repository¥connectorInstance
```

ビジネス・オブジェクト定義の作成

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

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

   ```
   ProductDir¥repository¥initialConnectorInstance
   ```

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

コネクター定義の作成

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

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

始動スクリプトの作成

始動スクリプトは以下のように作成します。

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

   dirname

2. この始動スクリプトを、26 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。

キューの Uniform Resource Identifiers (URI)

キューの URI は、シーケンス `queue://` で始まり、それに以下の項目が続きます。

- キューが存在しているキュー・マネージャーの名前
- もう 1 つの `/`
- キューの名前
- 残りのキュー・プロパティーを設定する、名前と値のペアのリスト (オプション)

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

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

表 11 に、キュー URI のプロパティー名を示します。

表 11. キューの URI に対する FIX 固有のコネクター・プロパティー名

プロパティー名	説明	値
`expiry`	ミリ秒で表した、メッセージの存続時間。	0 = 無制限。正整数 = タイムアウト (ミリ秒)。

表 11. キューの URI に対する FIX 固有のコネクター・プロパティー名 (続き)

プロパティー名	説明	値
priority	メッセージの優先順位。	0 から 9。1 が最高の優先順位。値 -1 は、このプロパティーがキューの構成によって決定されることを意味します。値 -2 は、コネクターが自分のデフォルト値を使用できることを示します。
persistence	メッセージをディスクに、「ハード化」するかどうか。	1 = 非永続。2 = 永続。値 -1 は、このプロパティーがキューの構成によって決定されることを意味します。値 -2 は、コネクターが自分のデフォルト値を使用できることを示します。
CCSID	宛先の文字セット。	整数。有効値は、WebSphere MQ の基本資料にリストされています。このプロパティーの値は、コネクター固有のプロパティー CCSID の値と一致している必要があります (23 ページの『CCSID』を参照)。
targetClient	受信アプリケーションが JMS 準拠であるかどうか。	0 = JMS (MQRFH2 ヘッダー)、1 = MQ (MQMD ヘッダー のみ)。
encoding	数値フィールドの表示方法。	WebSphere MQ の基本資料に記載されている整数値。

注: コネクターは、FIX メッセージ内のデータの文字セット (CCSID) またはエンコード属性を制御できません。データ変換は、データがメッセージ・バッファーから取り出されるか、あるいはメッセージ・バッファーに送達されるときに行われるため、コネクターは IBM WebSphere MQ にインプリメントされている JMS を使用してデータ変換を行います (IBM WebSphere MQ Java クライアント・ライブラリーの資料を参照してください)。したがって、これらの変換は、ネイティブの WebSphere MQ API がオプション MQGMO_CONVERT を使用して実行する変換と、双方向で同等でなければなりません。コネクターは、変換プロセスにおける差異または失敗を制御できません。コネクターは、特別な変更を必要とせずに、WebSphere MQ がサポートする、すべての CCSID またはエンコードのメッセージ・データを取り出すことができます。特定の CCSID またはエンコードのメッセージを送達するには、出力キューが完全修飾の URI で、CCSID と encoding の値を指定している必要があります。コネクターがこの情報を WebSphere MQ に渡し、WebSphere MQ は FIX を送達するためにデータをエンコードするときに (JMS API を介して) この情報を使用します。サポートされない CCSID およびエンコードについては、IBM の Web サイトから最新バージョンの IBM WebSphere MQ Java クライアント・ライブラリーをダウンロードすることによって解決できることが多いです。CCSID およびエンコー

ドに固有の問題が解決されない場合は、テクニカル・サポートに連絡して、代替の Java 仮想マシンを使用したコネクターの実行について相談してください。

メタオブジェクト属性構成

Connector for FIX Protocol は、以下の 2 種類のメタオブジェクトから処理に関する情報を取得できます。

- 静的なコネクター・メタオブジェクト
- 動的な子メタオブジェクト

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

静的メタオブジェクト

FIX Protocol の静的メタオブジェクトは、ビジネス・オブジェクトごとに定義された変換プロパティーのリストで構成されています。ビジネス・オブジェクトの変換プロパティーを定義するには、ストリング属性を作成し、構文 `busObj_verb` を使用してそれを命名します。例えば、動詞 Create を含む Customer オブジェクトの変換プロパティーを定義するには、`Customer_Create` という名前の属性を作成します。属性のアプリケーション固有テキストには、実際の変換プロパティーを指定します。

さらに、`Default` という名前の予約済み属性名を、メタオブジェクトに定義することもできます。この属性があると、そのプロパティーはすべてのビジネス・オブジェクトの変換プロパティーのデフォルト値として使用されます。

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

表 12 に、メタオブジェクトのプロパティーを示します。

表 12. 静的メタオブジェクト・プロパティー

プロパティー名	説明
CollaborationName	CollaborationName は、ビジネス・オブジェクトと動詞の組み合わせに対する属性の、アプリケーション固有テキスト内で指定される必要があります。例えば、ユーザーがビジネス・オブジェクト Customer に対する同期要求を動詞 Create で処理することを想定している場合、静的メタデータ・オブジェクトに Customer_Create という名前の属性が含まれている必要があります。Customer_Create 属性には、名前と値のペアを含む、アプリケーション固有テキストが入っていなければなりません。例えば、CollaborationName=MyCustomerProcessingCollab です。構文の詳細については、 32 ページの『アプリケーション固有のテキスト』のセクションを参照してください。この条件が満たされていない場合は、コネクターが Customer ビジネス・オブジェクトに関する要求を同期処理しようとすると、ランタイム・エラーが発生します。 **注:** このプロパティーは、同期要求にのみ有効です。
DataEncoding	DataEncoding は、メッセージの読み書きに使用されるエンコードです。このプロパティーが静的メタオブジェクトに指定されていない場合、コネクターは特定のエンコードを使用せずに、メッセージを読もうとします。動的な子メタオブジェクトに定義された DataEncoding は、静的なメタオブジェクトに定義された値をオーバーライドします。デフォルト値は Text です。この属性の値のフォーマットは、messageType[:*enc*] です。つまり、Text:ISO8859_1、Text:UnicodeLittle、Text、または Binary のようになります。
DataHandlerConfigMO	構成情報を提供するためにデータ・ハンドラーに渡されるメタオブジェクトです。静的なメタオブジェクトに指定された場合、このプロパティーは DataHandlerConfigMO コネクター・プロパティーに指定された値をオーバーライドします。さまざまなビジネス・オブジェクト・タイプを処理するために各種のデータ・ハンドラーが必要な場合は、この静的なメタオブジェクトのプロパティーを使用します。動的な子メタオブジェクトに指定された場合、このプロパティーはコネクター・プロパティーと静的なメタオブジェクトのプロパティーをオーバーライドします。データ形式が実際のビジネス・データに依存する可能性がある場合は、要求処理には動的な子メタオブジェクトを使用します。指定されたビジネス・オブジェクトはコネクターによりサポートされていることが必要です。

表 12. 静的メタオブジェクト・プロパティー (続き)

プロパティー名	説明
DataHandlerMimeType	特定の MIME タイプに基づいたデータ・ハンドラーを要求できます。静的なメタオブジェクトに指定された場合、この値は DataHandlerMimeType コネクター・プロパティーに指定された値をオーバーライドします。さまざまなビジネス・オブジェクト・タイプを処理するために各種のデータ・ハンドラーが必要な場合は、この静的なメタオブジェクトのプロパティーを使用します。動的な子メタオブジェクトに指定された場合、このプロパティーはコネクター・プロパティーと静的なメタオブジェクトのプロパティーをオーバーライドします。データ形式が実際のビジネス・データに依存する可能性がある場合は、要求処理には動的な子メタオブジェクトを使用します。DataHandlerConfigMO に指定されたビジネス・オブジェクトは、このプロパティーの値に対応する属性を含める必要があります。
DoNotReportBusObj	ユーザーは DoNotReportBusObj プロパティーを含めることができます。このプロパティーを true に設定すると、発行されるすべての PAN レポート・メッセージのメッセージ本体がブランクになります。要求側が要求が正常に処理されたことは確認したいが、ビジネス・オブジェクトへの変更に関する通知は必要としない場合に、このように設定することをお勧めします。このプロパティーは、NAN レポートには影響しません。静的メタオブジェクトでこのプロパティーが見つからない場合、コネクターはこれを false にデフォルト設定し、メッセージ・レポートにビジネス・オブジェクトを取り込みます。 **注:** このプロパティーは、同期要求にのみ有効です。
InputFormat	InputFormat は、特定のビジネス・オブジェクトと関連付けるメッセージ・フォーマットです。取り出されたメッセージがこのフォーマットである場合、メッセージは可能であれば特定のビジネス・オブジェクトに変換されます。デフォルトの変換プロパティーを使用する場合は、このプロパティーを設定しないでください。このプロパティーの値は、着信メッセージを、メッセージ内容の格納先になるビジネス・オブジェクトと対応付けるために使用されます。この機能は、Adapter for FIX Protocol では使用されません。
InputQueue	コネクター固有のプロパティーとしての InputQueue プロパティーは、アダプターのポーリング先キューを定義します。これは、アダプターでポーリング先キューの決定に使用される唯一のプロパティーです。静的メタオブジェクト内の InputQueue プロパティーは、アダプターでメッセージが特定のビジネス・オブジェクトにマップされる際に、InputFormat プロパティーとともに基準の役割を果たします。この機能は、Adapter for FIX Protocol では使用されません。
OutputFormat	OutputFormat は、指定されたビジネス・オブジェクトから作成されたメッセージで設定されます。OutputFormat が指定されていない場合、使用可能であれば入力フォーマットが使用されます。動的な子メタオブジェクトに定義された OutputFormat は、静的なメタオブジェクトに定義された値をオーバーライドします。

表 12. 静的メタオブジェクト・プロパティー (続き)

プロパティー名	説明
OutputQueue	OutputQueue は、特定のビジネス・オブジェクトから派生したメッセージが配信される出力キューです。動的な子メタオブジェクトに定義された OutputQueue は、静的なメタオブジェクトに定義された値をオーバーライドします。
ResponseTimeout	応答を待機した状態で、タイムアウトになるまでの時間をミリ秒で表します。このプロパティーが定義されていないかまたはゼロより小さい値が設定されている場合、コネクターは応答を待機せずにすぐに SUCCESS を戻します。動的な子メタオブジェクトに定義された ResponseTimeout は、静的なメタオブジェクトに定義された値をオーバーライドします。
TimeoutFatal	このプロパティーが定義されていて、値 True を含む場合、ResponseTimeout に指定された時間内に応答を受信しなければ、コネクターは APP_RESPONSE_TIMEOUT を戻します。応答メッセージを待機中のその他すべてのスレッドは、統合ブローカーにすぐに APP_RESPONSE_TIMEOUT を戻します。これにより、統合ブローカーはコネクターを終了します。動的な子メタオブジェクトに定義された TimeoutFatal は、静的なメタオブジェクトに定義された値をオーバーライドします。

表 13. Customer_Create のデフォルトの静的メタオブジェクト構造の例

属性名	アプリケーション固有のテキスト
Customer_Create	DataEncoding=Text:UnicodeLittle; OutputFormat=CUST_OUT; OutputQueue=QueueA; ResponseTimeout=10000; TimeoutFatal=False

アプリケーション固有のテキスト

アプリケーション固有のテキストは、名前と値のペア形式で構成され、それらはセミコロンで区切られています。例えば、次のようになります。

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

InputQueue へのデータ・ハンドラーのマッピング

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

1. コネクター固有プロパティー (24 ページの『InputQueue』を参照) を使用して、1 つ以上の入力キューを構成する。
2. それぞれの入力キューごとに、キュー・マネージャーおよび入力キュー名を指定し、またアプリケーション固有情報にデータ・ハンドラーのクラス名および MIME タイプを指定する。

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

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

入力フォーマットの多重定義

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

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

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

メタオブジェクトのサンプル

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

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

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

動詞 Create を含む Customer オブジェクト: 属性 `Customer_Create` は、コネクターに対して、フォーマットが `NEW` のメッセージについては動詞が Create でタイ

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

動詞 Update および Delete を含む Customer オブジェクト: 入力フォーマット MODIFY は多重定義されます。すなわち、動詞 Update を含む Customer ビジネス・オブジェクトと動詞 Delete を含む Customer ビジネス・オブジェクトの両方に定義されます。このフォーマットを持つ取り出されたメッセージを正常に処理するには、データ・ハンドラーが識別できるように、ビジネス・オブジェクト名と (該当する場合) 動詞をメッセージ内容に含める必要があります (33 ページの『入力フォーマットの多重定義』を参照してください)。要求処理操作では、出力フォーマットは定義されていないため、コネクターはどちらの動詞のメッセージも入力フォーマット MODIFY を使用して送信します。

動詞 Retrieve を含む Customer オブジェクト: 属性 Customer_Retrieve は、動詞 Retrieve を含むタイプ Customer のビジネス・オブジェクトが、フォーマット Retrieve を持つメッセージとして送信される必要があることを示します。デフォルトの応答時間は、コネクターがタイムアウトまでに最大 10000 ミリ秒待機するようにオーバーライドされていることに注意してください (それでも応答が受信されない場合には終了します)。

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

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

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

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

動的な子メタオブジェクト

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

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

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

表 13 および表 14 に、ビジネス・オブジェクト Customer_Create の静的なメタオブジェクトと動的な子メタオブジェクトのサンプルを示します。アプリケーション固有のテキストは名前と値のペアで構成され、それぞれのペアはセミコロンで区切られています。

表 14. Customer_Create の 動的な子メタオブジェクト構造

属性名	値
DataEncoding	Text:UnicodeLittle
DataHandlerMimeType[1]	fix
OutputFormat	CUST_OUT
OutputQueue	QueueA
ResponseTimeout	10000
TimeoutFatal	False

1. DataHandlerConfigMO は、コネクター構成プロパティーか静的なメタオブジェクトのいずれかに指定されていると想定します。

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

ポーリング中の動的な子メタオブジェクトの含まれるデータ

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

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

表 15. ポーリング用の JMS 動的子メタオブジェクト構造

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

表 15 に示すように、動的子メタオブジェクトで追加の属性 InputQueue を定義できます。この属性には特定のメッセージが取り出されるキューの名前が含まれます。子メタオブジェクト内にこのプロパティーが定義されていない場合、これらには値が取り込まれません。

シナリオ例:

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

動的な子メタオブジェクトのサンプル

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

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

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

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

[Attribute]
Name = FirstName
Type = String
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = LastName
Type = String
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
```

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

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

始動ファイルの構成

Connector for FIX Protocol を始動するには、始動ファイルを構成する必要があります。

Windows

Windows プラットフォームのコネクターの構成を完了するには、start_Fix.bat ファイルを修正する必要があります。

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

UNIX

UNIX プラットフォームのコネクターの構成を完了するには、start_Fix.sh ファイルを修正する必要があります。

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

始動

コネクターの始動と停止は、以下の説明に従って行います。

コネクターの始動

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

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 16 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 16. 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

この章では、FIX ビジネス・オブジェクトの構造に関してアダプターが前提として要求する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。

FIX ビジネス・オブジェクト構造の概要

FIX メッセージ対応のビジネス・オブジェクト定義は、Financial Information Exchange Protocol Organization から発行されている仕様に基づいています。包括的な情報については、以下のアドレスにある FIX (Financial Information Exchange) Protocol に関する資料を参照してください。

http://www.fixprotocol.org/cgi-bin/Welcome.cgi

ビジネス・オブジェクトと FIX メッセージの間のマッピングは明快です。トップレベル FIX ビジネス・オブジェクトの定義は、特定のバージョンの FIX (FIX 4.3 など) に基づく単一の FIX メッセージを表しています。FIX ビジネス・オブジェクトの命名規則と属性特性は、FIX メッセージの構造を反映しています。

以降のセクションでは、属性特性と命名規則について説明するほか、ビジネス・オブジェクト属性と FIX メッセージ・タグのマッピングがどのように行われるかについても説明します。

注: WebSphere では、FIX バージョン 4.0、4.1、4.2、および 4.3 のメッセージに対応するビジネス・オブジェクトをサポートしています。

コネクターはメタデータ主導型です。WebSphere ビジネス・オブジェクトでは、メタデータはアプリケーションに関するデータのことです。このデータはビジネス・オブジェクト定義に格納されており、コネクターとアプリケーションとの対話に役立ちます。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テキスト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、コネクターのコードを修正せずに、新規または変更済みのビジネス・オブジェクトを処理できます。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テキストの形式、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。そのため、FIX Protocol のビジネス・オブジェクトを作成または変更する場合、その変更内容は、コネクターが従うべきルールに適合している必要があります。適合していないと、コネクターは新規または変更済みのビジネス・オブジェクトを正しく処理できません。

標準命名規則

トップレベル・ビジネス・オブジェクトの命名規則は、次のとおりです。

FIX *<FIX_version>_<message_type_description>*

例えば、FIX43_ExecutionReport はバージョン 4.3 の FIX メッセージの名前です。

注: ビジネス・オブジェクト名の長さは、80 文字を超えてはいけません。

FIX ビジネス・オブジェクト定義に含まれる属性の名前は、いずれも固有の名前であることが必要です。属性と子ビジネス・オブジェクトの命名規則の詳細については、以降のセクションで説明します。

注: ビジネス・オブジェクト名と属性名は、Java の変数名の命名規則と同じ命名規則に従います。例えば、a から z、A から Z、_、および 0 から 9 までの文字に限り使用を許可されます。命名規則の詳細については、「*コンポーネント命名ガイド*」を参照してください。

ビジネス・オブジェクトとメッセージのマッピング

FIX メッセージは、いずれも、以下に示すような基本構造を備えています。

- 標準メッセージ・ヘッダー
- メッセージ・タグで構成された可変の本文テキスト
- 標準メッセージ・トレーラー

トップレベル FIX ビジネス・オブジェクトでは、この構造が以下のように反映されています。

- FIX 標準メッセージ・ヘッダーは、標準メッセージ・ヘッダー・タイプのコンテナー型子属性にマップされています (49 ページの『標準メッセージ・ヘッダー属性』を参照)。
- FIX 本文テキストは、単純型またはコンテナー型の子ビジネス・オブジェクト属性にマップされています (51 ページの『トップレベル・ビジネス・オブジェクトのサンプル』を参照)。
- FIX 標準メッセージ・トレーラーは、標準メッセージ・トレーラー・タイプのコンテナー型子属性にマップされています (50 ページの『標準メッセージ・トレーラー』を参照)。

注: どのビジネス・オブジェクト定義にも、少なくとも 1 つの属性がキー属性として定義されていなければなりません。FIX ビジネス・オブジェクトの定義では、最初の属性 (1 つ) がキー属性として定義されています (IsKey = true が設定されています)。

属性とタグのマッピング

ビジネス・オブジェクト属性のマッピングは、メッセージ・タイプについて定義している FIX 規格に基づいて行われます。(詳細については、http://www.fixprotocol.org/cgi-bin/Welcome.cgi にある FIX に関する資料を参照してください。)

FIX ビジネス・オブジェクト内の属性は、それぞれ、FIX メッセージ内のいずれかのタグにマップされています。一方、FIX メッセージ・タグは FIX Protocol に準拠しており、このプロトコルは以下の 2 つの構文でインプリメントされています。

- tag=value 構文
- FIXML 構文

Connector for FIX Protocol では、tag=value 構文を使用しているメッセージを処理します。

ビジネス・オブジェクト属性のマッピング規則は、FIX メッセージに使用される以下の種類のタグに対応しています。

- 単純タグ
- 繰り返しグループ
- コンポーネント (FIX バージョン 4.3 以降のみ)

これらの種類のタグに関しては、この後詳しく説明します。これらのタグは、FIX メッセージの標準メッセージ・ヘッダー、標準メッセージ・トレーラー、および本文テキストに出現します。

単純タグ

単純タグは、ビジネス・オブジェクト定義上、単純属性にマップされています。このタグは、次のように定義されている必要があります。

```
Name = <Name defined in FIX message definition>
Type = String
MaxLength = 255
IsKey = (see note below)
IsForeignKey = false
IsRequired = (see below)
AppSpecificInfo = (see below)
IsRequiredServerBound = false
[End]
```

注: 44 ページの『ビジネス・オブジェクトとメッセージのマッピング』の注を参照してください。

IsRequired

マップされる単純タグのいずれかが FIX のメッセージ・タイプ定義で必須タグとして定義されている場合、ビジネス・オブジェクト内の対応する属性を必須属性としてマークする必要があります。それ以外の属性については、この必須フラグを false に設定する必要があります。

AppSpecificInfo

単純タグに対応する属性のアプリケーション固有の情報 (AppSpecificInfo = に続く部分) には、以下の情報が含まれている必要があります。

- FIX タグ番号: TAG=<*タグ番号*> (例: TAG=8)
- データ型: TYPE=<*FIX メッセージ定義に定義されているデータ型*> (例: TYPE=String)
- 有効な値のリスト: VALUES= <*コンマで区切った値のリスト*> (例: VALUES=0,1,2)

  注: A から Z、0 から 3 などの値範囲はサポートされません。
- 名前と値のペアの間の区切りに使用する文字はセミコロンです (例: TAG=8;TYPE=String;VALUES=0,1,2)。

表 17 に示すのは、FIX によりデータ内容が検証される際に ISO コードまたは FIX で定義されている値セットが参照される FIX フィールドのリストです。これらのタグに関しては、FIX の型定義が IBM CrossWorlds の特定のデータ型値にマップされていなければなりません。

表 17. FIX と IBM CrossWorlds のデータ型の対応

FIX フィールド	FIX 属性タイプ	FIX バージョン	IBM CrossWorlds のデータ型値
ANY	Exchange	4.0 から 4.2	Exchange
ANY	Exchange/ ISO 10383	4.3	Exchange43
ANY	Char	4.0 または 4.1	String
SettlBrkrCode SettlInstCode SecuritySettlAgentCode CashSettlAgentCode	ISO 9362	4.3	ISO9362
CFICode		4.0 から 4.2	CFICode
CFICode	ISO 10962	4.3	CFICode43
Currency	ISO 4217	4.2 から 4.3	Currency
Country	ISO 3166	4.3	Country
SecurityID UnderlyingSecurityID	ISO 6166	4.3	ISO6166
SecurityType	任意	任意	SecurityType
YieldType	任意	任意	YieldType

注: FIX データ・ハンドラー用の子メタオブジェクトに含まれる属性の名前は、表 17 に示した IBM CrossWorlds のデータ型値と一致していなければなりません。データ・ハンドラー・メタオブジェクトの属性の詳細については、59 ページの表 19 を参照してください。

MultipleValueString

FIX バージョン 4.0 と 4.1 のメッセージでは、データ型が char のタグに複数の値を含めることができます。char 型として定義されており、かつ複数の値を含むことができるタグを正常に処理するためには、TYPE= に指定されているデータ型が、char ではなく MultipleValueString になっていることが必要です。

FIX データ型 MultipleValueString は、単純タグの特殊なケースです。この型は、複数の値を格納できるので、カーディナリティー n コンテナーのビジネス・オブジェクトとして定義される必要があります。このビジネス・オブジェクトは IBM CrossWorlds Exchange から入手することができ、その内容は以下のようになっています。

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

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

   [Verb]
   Name = Create
   [End]

   [Verb]
   Name = Delete
   [End]

   [Verb]
   Name = Retrieve
   [End]

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

型が MultipleValueString の FIX タグは、いずれも、ビジネス・オブジェクトの FIX_MultipleValueString 型のコンテナー属性にマップされていなければなりません。この規則が当てはまるのは、char として定義されていても複数の値をコンマで区切って格納することができる、バージョン 4.0 および 4.1 の FIX メッセージ・タグです。

繰り返しグループ

繰り返しグループは、1 つの FIX メッセージの中で複数回出現する可能性があるタグのセットです。繰り返しグループの正確な出現回数は、カウンターによって提示されます。FIX バージョン 4.3 では、このカウンターは必須であり、繰り返しグループの前に置かれていなければなりません。それ以前のバージョンの FIX では、繰り返しグループのカウンターは必須ではありませんが、ほとんどの繰り返しグループでカウンターが使用されています。

FIX ビジネス・オブジェクトの定義では、繰り返しグループは子ビジネス・オブジェクトとして定義されています。繰り返しグループには、別の繰り返しグループを含めることができます。FIX ビジネス・オブジェクトの定義でもこれと同じことが可能です。つまり、FIX ビジネス・オブジェクトには複数の子ビジネス・オブジェクトを含めることができ、子ビジネス・オブジェクトにはさらに複数の子ビジネス・オブジェクトを含めることができ、その後も同様に繰り返すことができます。

トップレベル・ビジネス・オブジェクト内の対応する子ビジネス・オブジェクト属性は、以下のように定義されている必要があります。

```
Name = <name of repeating group>
Type = <name of child business object definition>
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = N
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = (see below)
IsRequiredServerBound = false
```

繰り返しグループに必須タグが 1 つ以上含まれている場合、属性特性 AppSpecificInfo には以下の情報が含まれている必要があります。

- その繰り返しグループに含まれる最初の必須タグのタグ番号: TAG=<*タグ番号*>
- その繰り返しグループに対応するカウンター: CounterTag=<*その繰り返しグループの出現回数を示すカウンターのタグ番号*>

繰り返しグループに対応する属性に IsRequired = false が設定されている場合には、AppSpecificInfo に、その繰り返しグループに対応するカウンターが、CounterTag=<*その繰り返しグループの出現回数を示すカウンターのタグ番号*> の形式で含まれている必要があります。

名前と値のペアの間の区切りにはセミコロンを使用します。

対応する子ビジネス・オブジェクト (BO) 定義の命名規則は、次のとおりです。

<Top-level business object name>_RepGroup_<*group description*>

コンポーネント

FIX Protocol バージョン 4.3 では、メッセージ定義の中で、タグを論理的にグループ化したものとしてコンポーネントを定義しています。

ビジネス・オブジェクト定義上、コンポーネントは単一カーディナリティーの子オブジェクトにマップされています。

トップレベル・ビジネス・オブジェクト内の対応する子属性は、以下のように定義されている必要があります。

```
Name = <Component Name>
Type = <Name of child business object definition>
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = (see below)
AppSpecificInfo = (see below)
IsRequiredServerBound = false
```

注: コンポーネントに必須タグが 1 つ以上含まれている場合に限り、そのコンポーネントを必須としてマークする (IsRequired=true を設定する) ことができます。どのタグも明示的に必須タグとして定義されておらず、必須タグになるかどうかはビジネス記述次第であるような場合には、トップレベル・ビジネス・オブジェクト内の対応する属性の IsRequired 特性を true に設定しないでください。

コンポーネントに必須タグが 1 つ以上含まれている場合、対応する子属性の AppSpecificInfo には、最初の必須タグのタグ番号が次の形式で含まれていなければなりません。

```
TAG=<tag number>
```

対応する子ビジネス・オブジェクト定義の命名規則は、次のとおりです。

- FIX<FIX version>_ComponentBlock_<*group description*>

標準メッセージ・ヘッダー属性

トップレベル・ビジネス・オブジェクトの属性の 1 つは、FIX の標準メッセージ・ヘッダーにマップされています。この属性は、FIX ヘッダーが定義されている子ビジネス・オブジェクトを示します。

標準メッセージ・ヘッダー属性の命名規則は、次のとおりです。

FIX<*FIX_version*>_StandardMessageHeader

例: FIX43_StandardMessageHeader は、FIX バージョン 4.3 のメッセージ・ヘッダーにマップされている属性および子ビジネス・オブジェクトの名前です。

標準メッセージ・ヘッダーに対応する属性は、以下のように定義されている必要があります。

```
Name = Header
Type = FIX<FIX version>_StandardMessageHeader
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = (see below)
IsRequiredServerBound = false
```

アプリケーション固有の情報には、ストリング TYPE=HEADER が含まれている必要があります。このストリングは、この属性がヘッダー属性であることを示します。属性のタイプは、子ビジネス・オブジェクト定義の名前を示しています。定義のサンプルについては、52 ページの『メッセージ・ヘッダーに対応する子ビジネス・オブジェクトのサンプル』を参照してください。

注: 標準メッセージ・ヘッダーには、複数のタグを含めることができます。したがって、メッセージ・ヘッダー内のタグは、それぞれ、ヘッダーが定義されている子ビジネス・オブジェクトの属性にマップされている必要があります。この定義は、トップレベル・ビジネス・オブジェクトの標準メッセージ・ヘッダー属性から参照されています。

標準メッセージ・トレーラー

トップレベル・ビジネス・オブジェクトの属性の 1 つは、FIX の標準メッセージ・トレーラーにマップされています。この属性は、FIX トレーラーが定義されている子ビジネス・オブジェクトを示します。

標準メッセージ・トレーラー属性の命名規則は、次のとおりです。

FIX<*FIX_version*>_StandardMessageTrailer

例: FIX42_StandardMessageTrailer は、FIX バージョン 4.2 のメッセージ・トレーラーにマップされている属性および子ビジネス・オブジェクトの名前です。

標準メッセージ・トレーラーに対応する属性は、以下のように定義されている必要があります。

```
Name = Header
Type = <Name of BO definition for Standard Message Trailer>
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = (see below)
IsRequiredServerBound = false
```

属性のタイプは、子ビジネス・オブジェクト定義の名前を示しています。定義のサンプルについては、54 ページの『メッセージ・トレーラーに対応する子ビジネス・オブジェクトのサンプル』を参照してください。

注: 標準メッセージ・トレーラーには、複数のタグを含めることができます。したがって、メッセージ・トレーラー内のタグは、それぞれ、トレーラーが定義されている子ビジネス・オブジェクトの属性にマップされている必要があります。この定義は、トップレベル・ビジネス・オブジェクトの標準メッセージ・トレーラー属性から参照されています。

トップレベル・ビジネス・オブジェクトのサンプル

以下に示すのは、タイプ A (ログオン) の FIX メッセージに対応するトップレベル・ビジネス・オブジェクトの例です。最初の属性は FIX の標準メッセージ・ヘッダーにマップされており、最後の属性 (ObjectEventID 属性の直前の属性) は FIX の標準メッセージ・トレーラーにマップされている点に注意してください。FIX メッセージの本文テキストは、これらの間にある属性によって定義されています。

これらのヘッダーとトレーラーに対応する子ビジネス・オブジェクトの定義のサンプルについては、52 ページの『メッセージ・ヘッダーに対応する子ビジネス・オブジェクトのサンプル』および 54 ページの『メッセージ・トレーラーに対応する子ビジネス・オブジェクトのサンプル』を参照してください。

```
[BusinessObjectDefinition]
Name = FIX_Logon
Version = 3.0.0
AppSpecificInfo = TYPE=A

   [Attribute]
   Name = header
   Type = FIX_StandardMessageHeader
   ContainedObjectVersion = 1.0.0
   Relationship = Containment
   Cardinality = 1
   MaxLength = 0
   IsKey = false
   IsForeignKey = false
   IsRequired = true
   AppSpecificInfo =
   IsRequiredServerBound = false
   [End]

   [Attribute]
   Name = EncryptMethod
   Type = String
   MaxLength = 255
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = TAG=98;TYPE=Int;VALUES=0 , 1 , 2 , 3 , 4 , 5 , 6 ;
   IsRequiredServerBound = false
   [End]

....

   [Attribute]
   Name = Password
   Type = String
   MaxLength = 255
   IsKey = true
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = TAG=554
   IsRequiredServerBound = false
   [End]

   [Attribute]
   Name = trailer
   Type = FIX_StandardTrailer
   ContainedObjectVersion = 1.0.0
   Relationship = Containment
   Cardinality = 1
   MaxLength = 0
   IsKey = false
```

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

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

   [Verb]
   Name = Create
   [End]

   [Verb]
   Name = Delete
   [End]

   [Verb]
   Name = Retrieve
   [End]

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

メッセージ・ヘッダーに対応する子ビジネス・オブジェクトのサンプル

トップレベル・ビジネス・オブジェクトの標準メッセージ・ヘッダー属性は、1 つの子ビジネス・オブジェクトを示しています。以下に示すのは、そのようなビジネス・オブジェクトの定義のサンプルです。このサンプルでは、6 つのタグを含むヘッダーが定義されている点と、それらのタグのそれぞれに属性が用意されている点に注意してください。

```
[BusinessObjectDefinition]
Name = FIX43_StandardMessageHeader
Version = 3.0.0

[Attribute]
Name = BeginString
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = true
AppSpecificInfo = TAG=8; TYPE=String;VALUES=FIX.3.0, FIX.4.0,
FIX.4.1, FIX.4.2;
IsRequiredServerBound = false
[End]

[Attribute]
Name = BodyLength
Type = Integer
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
```

```
AppSpecificInfo = TAG=9 ;TYPE=Int;
IsRequiredServerBound = false
[End]

[Attribute]
Name = MsgType
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = true
AppSpecificInfo = TAG=35;TYPE=String;VALUES=$U, 0 , 1 , 2 ,
3 , 4 , 5 , 6 , 7 , 8 , 9 , A , B , C , D , E , F , G , H , J ,
K , L , M , N , P , Q , R , S , T , V , W , X , Y , Z , a , b ,
c , d , e , f , g , h , i , j , k , l , m ;
IsRequiredServerBound = false
[End]

...

[Attribute]
Name = OnBehalfOfSendingTime
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo= TAG=370; TYPE=UTCTimestamp;
IsRequiredServerBound = false
[End]

[Attribute]
Name = NoHops
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = TAG=627;
IsRequiredServerBound = false
[End]

[Attribute]
Name = HopCompID
Type = FIX_SMH_Hops
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = N
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = TAG=628;TagCounter=627
IsRequiredServerBound = false
[End]

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

[Verb]
```

```
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

メッセージ・トレーラーに対応する子ビジネス・オブジェクトのサンプル

トップレベル・ビジネス・オブジェクトの標準メッセージ・トレーラー属性は、1 つの子ビジネス・オブジェクトを示しています。以下に示すのは、そのようなビジネス・オブジェクトの定義のサンプルです。このサンプルでは、3 つのタグを含むトレーラーが定義されている点と、それらのタグのそれぞれに属性が用意されている点に注意してください。

```
[BusinessObjectDefinition]
Name = FIX43_StandardMessageTrailer
Version = 3.0.0

[Attribute]
Name = SignatureLength
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = TAG=93; TYPE=Int
IsRequiredServerBound = false
[End]

[Attribute]
Name = Signature
Type = LongText
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = TAG=89; TYPE=Data
IsRequiredServerBound = false
[End]

[Attribute]
Name = CheckSum
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = TAG=10; TYPE=String
IsRequiredServerBound = false
[End]

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

```
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

エラー処理

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

Message number
Message text

コネクターは、以下の各セクションで説明するような特定のエラーを処理します。

アプリケーションのタイムアウト

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

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

アンサブスクライブされたビジネス・オブジェクト

アンサブスクライブされたビジネス・オブジェクトに関連するメッセージを検索する場合、コネクターは、UnsubscribedQueue プロパティーに指定されたキューにメッセージを配信します。

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

データ・ハンドラーの変換

データ・ハンドラーがメッセージのビジネス・オブジェクトへの変換に失敗した場合、または、ビジネス・オブジェクトに特有の (JMS プロバイダーとは反対の) 処理エラーが発生した場合、メッセージは ErrorQueue に指定されたキューに配信されます。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・オブジェクトのメッセージへの変換に失敗した場合、BON_FAIL が戻されます。

トレース

トレースは、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、17 ページの『第 2 章 コネクターのインストール、構成、および始動』に記載されている『コネクター構成プロパティー』を参照してください。トレースの有効化および設定の方法など、詳細については、「*コネクター開発ガイド*」を参照してください。

以下に、コネクターのトレース・メッセージの推奨レベルを示します。

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

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

レベル 2 このレベルは、統合ブローカーにビジネス・オブジェクトが通知されるたびにログに記録されるトレース・メッセージに使用します。

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

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

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

第 4 章 FIX データ・ハンドラー

FIX データ・ハンドラーは、ビジネス・オブジェクトから FIX メッセージへ、および FIX メッセージからビジネス・オブジェクトへの変換を主な役割とするデータ変換モジュールです。デフォルトのトップレベル・データ・ハンドラー・メタオブジェクトのコネクターとサーバーはともに、固定の MIME タイプをサポートします。したがって、FIX データ・ハンドラーの使用もサポートします。

この章では、FIX データ・ハンドラーの構成方法について説明します。また、データ・ハンドラーによる FIX メッセージの処理方法についても説明します。

FIX データ・ハンドラーの構成

コネクターと使用するために FIX データ・ハンドラーを設定するには、以下の手順を実行する必要があります。

1. FIX データ・ハンドラーのクラス名が、コネクター・プロパティーとデータ・ハンドラー・メタオブジェクトに指定されていることを確認します。57 ページの『コネクター・メタオブジェクトの構成』を参照してください。
2. FIX データ・ハンドラー子メタオブジェクトの属性に対して、該当する値を入力します。 58 ページの『データ・ハンドラー子メタオブジェクトの構成』を参照してください。

注: FIX データ・ハンドラーが正常に機能するには、ビジネス・オブジェクト定義を、データ・ハンドラーをサポートするように作成または変更する必要があります。詳細については、 43 ページの『第 3 章 ビジネス・オブジェクト』を参照してください。

コネクター・メタオブジェクトの構成

FIX データ・ハンドラーと対話するようにコネクターを構成するには、コネクター固有プロパティー `DataHandlerClassName` の値が、`com.crossworlds.DataHandlers.fix.FixDataHandler` であることを確認します。

コネクターを動作させる前に、このプロパティーの値を設定する必要があります。この値を設定すると、FIX メッセージとビジネス・オブジェクト間で、双方向の変換を行うときに、コネクターが、FIX データ・ハンドラーにアクセスすることが可能になります。詳細については、 22 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・ハンドラー子メタオブジェクトの構成

WebSphere ソフトウェアには、FIX データ・ハンドラー用のデフォルト・メタオブジェクトの MO_DataHandler_Default_FIX が提供されています。このメタオブジェクトでは、型 MO_DataHandler_Fix の子属性が指定されています。表 18 に、子メタオブジェクト MO_DataHandler_FIX 内の属性を説明します。

表 18. FIX データ・ハンドラー用に提供されている子メタオブジェクト属性

属性名	説明	提供されているデフォルト値
ClassName	指定された MIME タイプで使用するためにロードするデータ・ハンドラー・クラスの名前。トップレベルのデータ・ハンドラー・メタオブジェクトの属性は、その名前が指定された MIME タイプと一致し、その型が FIX 子メタオブジェクトとなります。	com.crossworlds.DataHandlers.fix.FixDataHandler
DefaultVerb	ビジネス・オブジェクト作成時に使用される動詞。	Create
NumberOfEnvelopeHeaderField	先頭に追加された構文解析対象のメッセージ・エンベロープ・ヘッダーの数。この値は、tag=value[SOH] 区切り文字形式を取り、FIX 標準メッセージに追加する必要があります。[1]	0
SMH_BO_BASE_NAME	FIX メッセージ・ヘッダー・ビジネス・オブジェクト名を取得するために構文解析されます。FIX*nn* が、この名前の先頭に追加されます。例えば、FIX ヘッダーの FIX バージョン 4.3 ビジネス・オブジェクトは、FIX43_Standard_Message_Header となります。	StandardMessageHeader
ERROR_BO_BASE_NAME	エラー・レポート・ビジネス・オブジェクト名を取得するために構文解析されます。FIX*nn* が、この名前の先頭に追加されます。例えば、エラー・レポートの FIX バージョン 4.2 ビジネス・オブジェクト名は FIX42_ErrorReport となります。	ErrorReport
STATIC_LOOKUP_TYPES_LIST	このメタオブジェクトのコンマ区切り属性リストであり、ローカルに格納されている静的構成ファイルをポイントします。これらの構成ファイルは、コネクターと同じマシン上に常駐する必要があります。例えば、ファイル FIX43_MESSAGES には、FIX バージョン 4.3 用のメッセージ・タイプがすべて格納されます。STATIC_LOOKUP_TYPES_LIST 内の項目ごとに、別個の属性が、このデータ・ハンドラー子メタオブジェクト内に存在する必要があります。	FIX43_MESSAGES,FIX43_FIELDS
BO_NAMES	ビジネス・オブジェクト名の決定に使用される構成ファイルの名前。	<*pathname*>columns=4; file=bonames.cfg
FIXnn_MESSAGES	STATIC_LOOKUP_TYPES_LIST にリストされているローカル構成ファイルであり、FIX バージョン 4.0、4.1、4.2、および 4.3 用のメッセージ・タイプをすべて格納します。	<*pathname*>columns=1; file=FIX43_MessageTypes.cfg
FIXnn_FIELDS	STATIC_LOOKUP_TYPES_LIST にリストされているローカル構成ファイルであり、FIX バージョン 4.0、4.1、4.2、および 4.3 用の有効タグをすべて格納します。	<*pathname*>columns=1; file=FIX43_Tags.txt

表 18. FIX データ・ハンドラー用に提供されている子メタオブジェクト属性 (続き)

属性名	説明	提供されているデフォルト値
1. tag=value 構文と SOH 区切り文字の詳細については、FIX Protocol, Ltd. の資料を参照してください。		

表 18の「提供されているデフォルト値」列は、関連するメタオブジェクト属性のデフォルト値を示します。この子メタオブジェクトのすべての属性が、実際のシステムと FIX メッセージ・タイプに適切なデフォルト値を持っていることを確認する必要があります。また、少なくとも、ClassName 属性と DefaultVerb 属性がデフォルト値を持っていることを確認してください。

注: このメタオブジェクト内の属性にデフォルト値を割り当てるには、Business Object Designer を使用します。

WebSphere では、ISO コードまたは FIX 着信メッセージの検証を行うためにデータ・ハンドラーが使用する FIX 定義値セットに対応する属性を追加することを推奨しています。これらの属性の名前は、46 ページの『AppSpecificInfo』で説明されている属性に対する IBM CrossWorlds アプリケーション固有情報に定義されているデータ型に対応する必要があります。便宜のために、下の表 19 に、これらの属性の一部とその値を示します。

表 19. 追加のFIX データ・ハンドラー用子メタオブジェクト属性

属性名	説明	値
Exchange43	ISO 10383 コード・セットを格納しているローカル構成ファイルをポイントする属性。ISO 10383 コード・セットは、データ内容の検証を行うために使用されます。* この属性は、FIX バージョン 4.3 用であり、STATIC_LOOKUP_TYPES_LIST にリストしておく必要があります。	<*pathname*>columns=1; file=*filename*
Exchange	データ内容の検証に使用される FIX 定義値セットを格納しているローカル構成ファイルをポイントする属性。この属性は、FIX バージョン 4.0、4.1、および 4.2 用であり、STATIC_LOOKUP_TYPES_LIST にリストしておく必要があります。	<*pathname*>columns=1; file=*filename*
Currency	ISO 4217 通貨コードを格納しているローカル構成ファイルをポイントする属性。* この属性は、STATIC_LOOKUP_TYPES_LIST にリストしておく必要があります。	<*pathname*>columns=1; file=*filename*
Country	ISO 3166 国コード・データを格納しているローカル構成ファイルをポイントする属性。* この属性は、STATIC_LOOKUP_TYPES_LIST にリストしておく必要があります。	<*pathname*>columns=1; file=*filename*
ISO9362	ISO 9362 銀行 ID コード・データを格納しているローカル構成ファイルをポイントする属性。* この属性は、STATIC_LOOKUP_TYPES_LIST にリストしておく必要があります。	<*pathname*>columns=1; file=*filename*

表 19. 追加のFIX データ・ハンドラー用子メタオブジェクト属性 (続き)

属性名	説明	値
CFICode43	ISO 10962 金融商品種別データを格納しているローカル構成ファイルをポイントする属性。* この属性は、FIX バージョン 4.3 用であり、STATIC_LOOKUP_TYPES_LIST にリストしておく必要があります。	<*pathname*>columns=1; file=*filename*
CFICode	金融商品種別データを格納しているローカル構成ファイルをポイントする属性。この属性は、FIX バーション 4.0、4.1、および 4.2 用であり、STATIC_LOOKUP_TYPES_LIST にリストしておく必要があります。	<*pathname*>columns=1; file=*filename*
ISO6166	ISO 6166 国際セキュリティー ID データを格納しているローカル構成ファイルをポイントする属性。* この属性は、STATIC_LOOKUP_TYPES_LIST にリストしておく必要があります。	<*pathname*>columns=1; file=*filename*

* 上記のコードを取得するには、ISO の www.ISO.ch に連絡する必要があります。取得したら、それらのコードを格納するファイルを作成し、データ・ハンドラー・メタオブジェクト内の子属性である STATIC_LOOKUP_TYPES_LIST を定義するコンマ区切りリストに属性名を追加する必要があります。

属性を追加するには、以下の手順を実行します。

1. ISO ファイルをダウンロードして属性にポイントさせ、ファイルのコンテンツを使用して、ローカル構成ファイルを作成します。
2. Business Object Designer を使用して、データ・ハンドラー子メタオブジェクト MO_DataHandler_FIX に属性を追加し、値を割り当てます。値は、ステップ 1 の ISO コードなどを格納するファイルの名前となります。子属性名は、表 19 の例示に従う必要がありますが、属性がポイントする情報を格納するファイルに対しては、任意の名前を選択することができます。ファイル名に加えて、絶対パス名も指定する必要があります。例えば、属性 Currency は、C:¥dependencies¥Currency.cfg ファイルをポイントします。
3. データ・ハンドラー・メタオブジェクト内のコンマ区切り子属性である STATIC_LOOKUP_TYPES_LIST に、ローカル構成ファイルをポイントする属性名を追加します (表 18 を参照)。

注: FIX データ・ハンドラーは、FIX 発信メッセージ作成の場合にも、MO_DataHandler_FIX に定義されている属性をチェックします。

ビジネス・オブジェクトの要件

FIX データ・ハンドラーは、ビジネス・オブジェクトまたは FIX メッセージを変換するときにビジネス・オブジェクト定義を使用します。変換は、ビジネス・オブジェクトの構造とそのアプリケーション固有テキストを使用して実行されます。FIX データ・ハンドラーの要件に、ビジネス・オブジェクト定義が準拠することを確実にするには、43 ページの『第 3 章 ビジネス・オブジェクト』に説明されている指針に従います。

FIX メッセージへのビジネス・オブジェクトの変換

ビジネス・オブジェクトを FIX メッセージに変換するため、FIX データ・ハンドラーは、トップレベルのビジネス・オブジェクト内の属性を順番にループします。データ・ハンドラーは、属性が必要であるかどうかに関係なく、データが入力されている属性をすべて構文解析し、ビジネス・オブジェクトとその子オブジェクト内で属性が出現する順序に従って、FIX メッセージのブロックを再帰的に生成します。複数値ストリングは、スペースを使用して連結されます (複数値ストリングの詳細については、47 ページの『MultipleValueString』を参照)。

注: ビジネス・オブジェクトの FIX メッセージへの変換時、データ・ハンドラーはデータ検証を実行しません。

ビジネス・オブジェクトへの FIX メッセージの変換

FIX データ・ハンドラーは、ビジネス・オブジェクトの構成と準拠性に対してだけでなく、FIX メッセージのフォーマットと構文に対する検証も行います。エラー・レポートおよび検証の詳細については、63 ページの『エラー処理』を参照してください。

注: 検証は、ビジネス・オブジェクト定義を優先して行われます。ビジネス・オブジェクト定義が正しくない場合、FIX 着信メッセージは、その構文やフォーマットが正しくても、拒否されることがあります。

以下に説明するように、FIX データ・ハンドラーは、FIX メッセージからデータを抽出し、ビジネス・オブジェクト内に対応する属性を設定します。

メッセージの構文解析

データ・ハンドラーは、着信データ・ストリームを tag=value フィールドに構文解析します。tag=value フィールドは、非印刷 ASCII 文字 SOH (Start Of Header) によって区切られますが、型 `data` の tag=value フィールドは例外であり、その他の埋め込み区切り文字が許されます。したがって、データ・ハンドラーは、tag=value ペアの 型 `data` とその他すべての 2 種類を区別します。型 `data` のデータは、長さパラメーターを使用して構文解析されます。その他すべての型のタグ・フィールドは、SOH 区切り文字を使用して構文解析されます。型 `data` のタグと区切り規則の詳細については、FIX の資料を参照してください。

注: 型情報は、構文解析対象の FIX メッセージに対応するビジネス・オブジェクト定義のアプリケーション固有テキストに格納されています。

メッセージ・ヘッダー・ビジネス・オブジェクト名の決定

ヘッダーは、FIX メッセージ内の BeginString (タグ 8) を使用して、FIX メッセージのバージョンを決定します。バージョンは、標準メッセージ・ヘッダー・ビジネス・オブジェクトの名前の先頭に付けられます。例えば、FIX タグ `8=4.3` は、FIX バージョン 4.3 を示します。データ・ハンドラーは、`4.3` を `43` に変換した後、ヘッダー構成メタオブジェクト属性を使用して、ビジネス・オブジェクトの名前を決定し、メッセージ・ヘッダーを表します (`FIX43_StandardMessageHeader`)。次にデータ・ハンドラーは、メッセージ・ヘッダー・ビジネス・オブジェクトを使用して、FIX メッセージのフォーマットを tag=value フォーマットまたは FIXML フォ

ーマットのいずれにするかを決定します (Connector for FIX は、FIXML フォーマットの FIX メッセージをサポートしません)。

FIX 標準メッセージ・ヘッダーが構文解析できない場合は、MalFormDataException がスローされます。ただし、メッセージ・ヘッダーの処理中に FIXException がスローされた場合、データ・ハンドラーは、エラーのフラグを立てますが、ヘッダーが検証を通過するまで、またはデータ・ストリームが尽きるまでメッセージの構文解析を続行します。データ・ハンドラーは、ErrorReport ビジネス・オブジェクトの作成を支援するため処理を続行します。ErrorReport ビジネス・オブジェクトの詳細については、63 ページの『検証エラー』を参照してください。

トップレベル・ビジネス・オブジェクト名の決定

標準メッセージ・ヘッダーの構文解析を完了すると、データ・ハンドラーは FIX バージョン番号を使用して、データ・ハンドラー・メタオブジェクト属性 BO_NAMES がポイントするローカル構成ファイルの bonames.cfg を読み取ります。bonames.cfg ファイルには、サポートされているビジネス・オブジェクトの名前がすべて格納されています。データ・ハンドラーは、メッセージ・タイプとバージョンを使用して、bonames.cfg ファイル内のトップレベル・ビジネス・オブジェクト名を発見します。例えば、bonames.cfg ファイル内の以下の項目には、型 ExecutionReport のバージョン 4.3 FIX メッセージに関連する名前 FIX43_ExecutionReport がリストされています。

```
FIX43   8   N   FIX43_ExecutionReport
```

ここで、N は FIXML オブジェクトでないことを表します。

注: bonames.cfg ファイルには、すべてのサポートされているビジネス・オブジェクトの名前、それに対応するバージョンとメッセージ・タイプを追加する必要があります。

FIX タグからビジネス・オブジェクト属性へのマッピング

データ・ハンドラーは、FIX 構文規則を使用して、メッセージ・タグをビジネス・オブジェクト属性にマップします。特に、データ・ハンドラーは、属性アプリケーション固有テキスト・プロパティー TAG=*nn* を使用して、ビジネス・オブジェクト属性に FIX タグをマップします。この属性プロパティーの詳細については、45 ページの『属性とタグのマッピング』を参照してください。

ビジネス・オブジェクト属性値の設定時には、以下の規則が遵守されます。

1. タグの属性が単純属性である場合、属性の値には、その型の値がセットされます。単純属性の詳細については、45 ページの『単純タグ』を参照してください。
2. 属性が型 MultipleValueString でなく、カーディナリティー n コンテナーである場合、データ・ハンドラーは、現在のタグが、ある繰り返しグループの各インスタンスの先頭を示すものと推定します。繰り返しグループの場合、データ・ハンドラーは、属性の順序に対して検証を行います。繰り返しグループの詳細については、48 ページの『繰り返しグループ』を参照してください。

   注: 繰り返しグループの各インスタンス内のタグは、同じ順序をとる必要があります。

3. 属性が型 MultipleValueString で、カーディナリティー n コンテナーである場合、値は、「 」(スペース) を区切り文字として使用し構文解析されます。詳細については、 47 ページの『MultipleValueString』を参照してください。

   注: データ・ハンドラーにとって、繰り返しグループとその他すべての子ビジネス・オブジェクトの違いは、属性の順序に対して検証を行う必要があるかという点です。

   上記以外の場合において属性が単一カーディナリティー・コンテナーであるとき、その属性はコンポーネントを表し、再帰的に処理されます。コンポーネントが繰り返しグループ内に存在する場合、属性の順序が維持されます。詳細については、 48 ページの『コンポーネント』を参照してください。
4. データ・ハンドラーは、ビジネス・オブジェクト全体にわたってタグに一致する属性を探索します。タグに一致する属性が現在のビジネス・オブジェクト内に発見されなかった場合、再帰処理が停止してメソッドが復帰します。タグがいずれのビジネス・オブジェクト内にも発見されなかった場合、エラーが登録されます。

エラー処理

FIX データ・ハンドラーは、以下で説明されているように、構成、I/O、およびデータ検証の各エラーに対応します。

構成エラー

データ・ハンドラー子メタオブジェクトに必須の値が欠けている場合、または FIX ビジネス・オブジェクトが、事前定義された構造に従っていない場合、データ・ハンドラーは以下の例外をスローします。

```
com.crossworlds.DataHandlers.Exceptions.ConfigurationException
```

この例外は特に、属性アプリケーション固有テキスト内の型プロパティーが、サポートされている FIX メッセージ・タイプでなく、いずれの STATIC_LOOKUP_TYPES_LIST 構成ファイルにも登録されていないことを意味します。(58 ページの『データ・ハンドラー子メタオブジェクトの構成』を参照。) コネクターが、この例外をキャッチし、致命的エラーとしてログに記録します。

I/O エラー

java.io クラスのいずれかが IOException をスローした場合はすべて、致命的エラーが発生します。

検証エラー

データ・ハンドラーは、FIX メッセージからビジネス・オブジェクトへの処理時にデータ検証エラーを検出します。データ検証エラーは、データ・ハンドラーがメッセージ・ヘッダー・ビジネス・オブジェクトに正常にデータをセットした後に検出されます。データ・ハンドラーは、メッセージ・ヘッダー情報を利用して、エラー・レポートを作成します。ただし、データが正常にメッセージ・ヘッダー・ビジネス・オブジェクトにセットされず、検証を通過しなかった場合は、例外の com.crossworlds.DataHandlers.Exceptions.MalformedDataException がスローされます。

これは、致命的エラーです。

FIX メッセージ・データの検証は、以下の内容と比較して行われます。

- 対応するビジネス・オブジェクト定義のアプリケーション固有テキスト。
- データ・ハンドラー子メタオブジェクトの STATIC_LOOKUP_TYPES_LIST 属性にリストされている構成ファイルの内容。(58 ページの『データ・ハンドラー子メタオブジェクトの構成』を参照。) 例えば、FIX バージョン 4.3 メッセージのメッセージ・タイプに対する検証は、FIX43_MESSAGES ファイルに格納されているメッセージ・タイプと比較して行われます。このメッセージ内のタグは、FIX43_FIELDS ファイルに格納されているタグと比較され、以降も同様です。

データ検証エラーは、エラー・レポート・ビジネス・オブジェクトを通して報告されます。エラー・レポート・ビジネス・オブジェクトは、¥samples ディレクトリー内に格納され出荷されています。このオブジェクトは、FIX*nn* _ErrorReport という形式を取ります。ここで、*nn* は、FIX リリース・バージョンです。
(「ErrorReport」は、データ・ハンドラー子メタオブジェクトに指定されている値であり、変更可能です。58 ページの『データ・ハンドラー子メタオブジェクトの構成』を参照してください。)

注: データ検証エラー・レポートが正常に機能するには、FIX *nn* _ErrorReport ビジネス・オブジェクトにサブスクライブするビジネス・プロセス (ICS アダプター用のコラボレーション) を作成する必要があります。作成したビジネス・プロセスでは、このビジネス・オブジェクトをコネクターから取得し、それを拒否メッセージに変換した後、拒否メッセージを FIX メッセージの送信側に送信します。統合ブローカーとして ICS を使用するときに、上記のようなコラボレーションを作成する場合は、「*コラボレーション開発ガイド*」を参照してください。

表 20 に、エラー・レポート・ビジネス・オブジェクトの構造を示します。

表 20. エラー・レポート・ビジネス・オブジェクトの構造

属性名	属性タイプ	コメント
Header	FIX*nn*_StandardMessageHeaderCardinality = 1	検証に失敗したメッセージのヘッダー
ErrorMsg	String	FIX エラー・メッセージ
ErrorCode	String	FIX エラー・コード
ErrorTag	String	検証に失敗した FIX フィールドの番号

FIX エラー・コード

以下の各セクションでは、エラー・コード・ビジネス・オブジェクト内に戻される ErrorCode 値を説明します。FIX エラーの詳細については、以下から入手できる FIX Protocol (Financial Information Exchange Protocol) の資料を参照してください。

http://www.fixprotocol.org/cgi-bin/Welcome.cgi

エラーは、データ・ハンドラーによって処理されます。表 21 は、これらのエラーを要約したものです。

注: エラー・コード値 12 以上は、FIX バージョン 4.3 のみに適用されます。FIX 4.1 では、エラー・メッセージ・テキストが定義されており、数値エラー・コードを使用していません。

表 21. FIX エラー・コード

エラー・コード	説明
0	無効なタグ番号です。FIX パーサーがチェックします。
1	必須のタグが欠落しています。ビジネス・オブジェクトのマッピング後、データ・ハンドラーが、isRequired 属性プロパティーを使用してチェックします。
2	メッセージ・タイプにとって未定義のタグです。ビジネス・オブジェクトのマッピング時に、データ・ハンドラーがチェックします。
3	メッセージ・バージョンにとって未定義のタグです。データ・ハンドラーが、ErrorCode 2 の検出時に、このチェックを行います。
4	値なしに指定されたタグです。FIX パーサーが tag=[SOH] として検出します。
5	このタグにとって、値が正しくありません (範囲外)。データ・ハンドラーが、ビジネス・オブジェクト定義の VALUES= *<コンマ区切りリスト>* アプリケーション固有テキスト・プロパティーにリストされている値とこの値を比較チェックします。データ・ハンドラーは、データ・ハンドラー子メタオブジェクト属性がポイントするファイル内の値とこの値の比較チェックも行います。
6	値のデータ・フォーマットが正しくありません。FIX では、string と custom を除いて、すべてのフォーマット (int、float、char、UTCDatetime など) に対して検証を行うように規定されています。型 custom のデータは、データ・ハンドラー子メタオブジェクトの STATIC_LOOKUP_TYPES_LIST 属性にリストされている型と比較されます。データ・フォーマットが検証に適合しない場合は、構成例外がスローされます。
7	N.A.
8	
9	
10	
11	メッセージ・タイプが無効です。データ・ハンドラーが、有効な値を格納しているルックアップ・ファイルをポイントするデータ・ハンドラー子メタオブジェクト内の FIX*nn*Messages 属性を使用して検出します。
12	XML 検証エラーです。サポートされていません。
13	タグが複数回出現しています。このエラーは、データが繰り返しグループのものである場合には発生しません。
14	指定されたタグが必要な順序に並んでいません。
15	繰り返しグループのフィールドの順序が正しくありません。
16	繰り返しグループの NumInGroup カウントが正しくありません。NumInGroup 値が、対応するコンテナー内のビジネス・オブジェクトの数と一致しません。
17	非「data」値に、フィールド区切り文字 (SOH 文字) が含まれています。

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実行時に検出される問題について説明します。

始動時の問題

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。start_connector.bat の変数 MQSERIES_JAVA_LIB が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認してください。
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。start_connector.bat の変数 MQSERIES_JAVA_LIB が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認してください。
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。start_connector.bat の変数 MQSERIES_JAVA_LIB が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認してください。
コネクターが初期設定中に不意にシャットダウンして、「java.lang.UnsatisfiedLinkError: no mqjbnd01 in shared library path 」という例外がレポートされた。	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要なランタイム・ライブラリー (mqjbnd01.dll [NT] または libmqjbnd01.so [Solaris]) を見つけることができません。パスに IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーが含まれているか確認してください。
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

問題	考えられる解決策と説明
コネクターが、MQRFH2 ヘッダーの付いたすべてのメッセージを送達する。	MQMD WebSphere MQ ヘッダーの付いたメッセージのみを送達するには、?targetClient=1 を出力キュー URI の名前に追加します。例えば、メッセージをキュー queue://my.queue.manager/OUT に出力するには、URI を queue://my.queue.manager/OUT?targetClient=1 に変更します。詳しくは、17 ページの『第 2 章 コネクターのインストール、構成、および始動』を参照してください。
コネクターが、コネクター・メタオブジェクトでのフォーマットの定義に関係なく、すべてのメッセージ・フォーマットを送達時に 8 文字に切り捨てる。	これは、コネクターではなく、WebSphere MQ MQMD メッセージ・ヘッダーの制約です。

付録 A. コネクターの標準構成プロパティー

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

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

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

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

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

新規プロパティーと削除されたプロパティー

以下の標準プロパティーは、本リリースで追加されました。

新規プロパティー

- XMLNameSpaceFormat

削除されたプロパティー

- RestartCount
- RHF2MessageDomain

標準コネクター・プロパティーの構成

Adapter コネクターには 2 つのタイプの構成プロパティーがあります。

- 標準構成プロパティー
- コネクター固有の構成プロパティー

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

Connector Configurator の使用

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

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

プロパティー値の設定と更新

プロパティー・フィールドのデフォルトの長さは 255 文字です。

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

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

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

- **動的**
  変更を System Manager に保管すると、変更が即時に有効になります。コネクターが System Manager から独立してスタンドアロン・モードで稼働している場合 (例えば、いずれかの WebSphere Message Brokers と連携している場合) は、構成ファイルでのみプロパティーを変更できます。この場合、動的更新は実行できません。
- 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 22 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 22. 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 22. 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の場合は 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 22. 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: <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 22. 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	
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します。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。

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

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

- RepositoryDirectory がローカル・ディレクトリーの場合は、設定は CwXML になります。
- RepositoryDirectory の値が <REMOTE> の場合には、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・パスに円記号 (¥) を使用します。UNIX システムを使用している場合は、円記号をスラッシュ (/) に置き換えてください。また、各オペレーティング・システムの規則に従ってください。

この付録では、次のトピックについて説明します。

Connector Configurator の概要

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

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

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

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

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

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

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

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

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

ただし**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクター固有プロパティーのテンプレートを作成します。システム内ですで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、89 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

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

Connector Configurator の始動

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

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

スタンドアロン・モードでの Configurator の実行

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

これを行うには、以下のステップを実行します。

- **「スタート」>「プログラム」**から、**「IBM WebSphere InterChange Server」>「IBM WebSphere Business Integration 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: Integration Broker」**の隣のプルダウン・メニューをクリックします。使用しているブローカーに応じて、ICS、WebSphere Message Brokers、または WAS を選択します。

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (94 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。
- **「Integration Broker」**フィールドで、ICS 接続、WebSphere Message Brokers 接続、WAS 接続のいずれかを選択します。
- この付録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. この付録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、この付録で後述するように、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 97 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、以下のステップを実行します。

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

アプリケーション固有の構成プロパティーの設定

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

1. グリッドの左上端の部分で右マウス・ボタンをクリックします。ポップアップ・メニュー・バーが表示されます。プロパティーを追加するときは**「追加」**をクリックします。子プロパティーを追加するには、親の行番号で右マウス・ボタンをクリックし、**「子を追加」**をクリックします。
2. プロパティーまたは子プロパティーの値を入力します。
3. プロパティーを暗号化するには、**「暗号化」**ボックスを選択します。
4. 96 ページの『標準コネクター・プロパティーの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

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

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

コネクター・プロパティーの暗号化

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

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

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

更新メソッド

付録『コネクターの*標準構成プロパティー*』の 70 ページの『プロパティー値の設定と更新』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

コネクターで使用するビジネス・オブジェクトを指定するには、Connector Configurator の**「サポートされているビジネス・オブジェクト」**タブを使用します。

汎用ビジネス・オブジェクトと、アプリケーション固有のビジネス・オブジェクトの両方を指定する必要があり、またそれらのビジネス・オブジェクト間のマップの関連を指定することが必要です。

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

ご使用のブローカーが ICS の場合

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

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

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

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

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

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

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

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

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

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

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

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

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

スタンドアロン・モードで作業している (System Manager に接続していない) 場合、手動でビジネス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。

以下のをステップを実行して、マップを明示的にバインドします。

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

リソース (ICS)

「リソース」タブでは、コネクター・エージェントが、コネクター・エージェント並列処理を使用して同時に複数のプロセスを処理するかどうか、またどの程度処理するかを決定する値を設定できます。
すべてのコネクターがこの機能をサポートしているわけではありません。複数のプロセスを使用するよりも複数のスレッドを使用する方が通常は効率的であるため、Java でマルチスレッドとして設計されたコネクター・エージェントを実行している場合、この機能を使用することはお勧めできません。

メッセージング (ICS)

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

トレース/ログ・ファイル値の設定

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

ログとトレースの値を変更するには、以下のステップを実行します。

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

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

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

データ・ハンドラー

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

これらのプロパティーに使用する値については、付録 A の『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、以下のステップを実行します (オプション)。

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

構成の完了

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

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

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

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

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

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

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

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

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

```
                                <Value>es_ES</Value>
                                <Value>pt_BR</Value>
                                <Value>en_US</Value>
                                <Value>en_GB</Value>
                        <DefaultValue>en_US</DefaultValue>
                </ValidValues>
        </Property>
```

付録 C. 特記事項

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

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

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

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

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

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

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

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

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

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

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

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

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

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

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

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

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 V 2.4.0