WebSphere®

IBM

Adapter for TCP/IP ユーザーズ・ガイド

WebSphere®

IBM

Adapter for TCP/IP ユーザーズ・ガイド

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、 79 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere Business Server Express および Express Plus Adapter for TCP/IP (5724-i47) バージョン 1.0.0 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
Adapter for TCP/IP User Guide

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 2005.9

この文書では、平成明朝体™W3、平成明朝体™W7、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™W7、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

本書について

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

IBM(R) WebSphere(R) Business Integration Adapters 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・システムおよびメインフレーム・システムに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 Server Express および Express Plus Adapter for TCP/IP のインストール、構成、トラブルシューティング、およびビジネス・オブジェクト開発について説明します。

特に明記されていない限り、本書のすべての情報は、IBM WebSphere Business Integration Server Express および IBM WebSphere Business Integration Server Express Plus の両方に適用されます。「WebSphere Business Integration Server Express」という用語とその変形は、両方の製品を指します。

対象読者

本書は、お客様のサイトでアダプターを使用するコンサルタント、開発者、およびシステム管理者を対象としています。

関連文書

この製品に付属する資料の完全セットで、すべての WebSphere Business Integration Server Express 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。

関連文書は、以下のサイトでダウンロード、インストール、および表示することができます。

- http:/www.ibm.com/websphere/wbiserverexpress/infocenter

表記上の規則

この資料は下記の規則に従って編集されています。

Courier フォント	コマンド名、ファイル名、入力情報、システムが画面に出力した情報などのリテラル値を示します。
イタリック 青のアウトライン	初出語、変数名、または相互参照を示します。 青のアウトラインは、マニュアルをオンラインで表示するときにのみ見られるもので、相互参照用のハイパーリンクを示します。アウトラインの内側をクリックすることにより、参照先オブジェクトにジャンプできます。
{ }	構文の説明で、複数のオプションが中括弧で囲まれている場合は、その中の 1 つのオプションのみを選択することが必要です。
|	構文の記述行の場合、パイプ記号 (|) は、選択対象のオプションの区切りです。1 つのオプションのみを選択する必要があります。
[ ]	構文の記述行で、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。
. . .	構文の説明で、省略符号は、直前のパラメーターの繰り返しを示します。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示します。
< >	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・パスの規則として円記号 (¥) を使用します。Linux システムの場合には、円記号をスラッシュ (/) に置き換えてください。すべての製品パス名は、マシン上でコネクターがインストールされているディレクトリーを基準とした相対パス名です。
ProductDir	IBM WebSphere Business Integration Server Express および Express Plus アダプター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

バージョン 1.0.0 は、WebSphere Business Integration Server Express Adapter for TCP/IP の最初のリリースです。

第 1 章 概要

- 『作業ロードマップ』
- 『用語』
- 3 ページの『コネクター・アーキテクチャー』

この章では、TCP/IP アダプターの概要を簡単に説明し、理解する必要がある用語を説明して、アダプターの処理について説明します。アダプターのインストール、構成、および使用の前に、アダプターに関する基本的な知識を身につけることが重要です。

作業ロードマップ

Adapter for TCP/IP を使用するには、表 1 に説明されている作業を実行する必要があります。

表 1. アダプターの使用: 作業ロードマップ

作業	関連手順 (参照先)	詳細情報 (参照先)
アダプターのインストール	5 ページの『第 2 章 コネクターのインストール』	「*WebSphere Business Integration Server Express インストール・ガイド (Windows 版)*」、「*WebSphere Business Integration Server Express インストール・ガイド (Linux 版)*」、または「*WebSphere Business Integration Server Express インストール・ガイド (OS/400 版)*」
ビジネス・オブジェクトとメタオブジェクトの構成	9 ページの『第 3 章 TCP/IP アダプターのビジネス・オブジェクトおよびメタオブジェクト』	「*ビジネス・オブジェクト開発ガイド*」
コネクターの構成	19 ページの『第 4 章 コネクターの構成』 45 ページの『付録 A. 標準構成プロパティー』 67 ページの『付録 B. コネクター固有のプロパティーおよび必須のビジネス・オブジェクト・プロパティー』	「*コネクター開発ガイド*」
コネクターの実行	35 ページの『第 5 章 コネクターの実行』	「*コネクター開発ガイド*」
コネクターの保守	41 ページの『第 6 章 コネクターの保守』	

用語

アダプターを理解するには、次の用語を理解する必要があります。

アダプター

統合ブローカーとアプリケーション (またはテクノロジー) との間の通信をサポートするコンポーネントを備えている、WebSphere Business Integration システムのコンポーネント。アダプターには、コネクター、メッセージ・ファイル、および構成ツールが必ず含まれています。また、Object Discovery Agent (ODA) やデータ・ハンドラーも含まれていることがあります。

アダプター・フレームワーク

IBM が提供する、アダプターの構成と実行のためのソフトウェア。アダプター・フレームワークには、ランタイム・コンポーネントとして、Java ランタイム環境、コネクター・フレームワーク、および Object Discovery Agent (ODA) ランタイムが組み込まれています。コネクター・フレームワークには、コネクターを新規開発するときに必要となるコネクター・ライブラリー (C++ および Java) が含まれています。ODA ランタイムには、ODA を新規開発するときに必要となる Object Development Kit (ODK) のライブラリーが含まれています。構成用コンポーネントとしては、以下のツールが用意されています。

- Business Object Designer Express
- Connector Configurator Express
- Log Viewer
- System Manager
- System Monitor
- Test Connector
- アダプターに関連付けられた Object Discovery Agent (ODA) (用意されていない場合もあります)

Adapter Development Kit (ADK)

アダプター開発用のサンプルをいくつか備えた開発キット。サンプルには、コネクターと Object Discovery Agent (ODA) のサンプルも含まれます。

コネクター

ビジネス・オブジェクトを使用して、統合ブローカーにイベント関連の情報を送信し (イベント通知)、統合ブローカーから要求関連の情報を受信する (要求処理)、アダプターのコンポーネント。コネクターは、コネクター・フレームワークとコネクターのアプリケーション (またはテクノロジー) 固有コンポーネントから構成されます。

コネクター・フレームワーク

コネクターのアプリケーション (またはテクノロジー) 固有コンポーネントと統合ブローカーとの対話を管理するコネクターのコンポーネント。このコンポーネントは、必要な管理サービスをすべて備えており、コネクターが必要とするメタデータをリポジトリーから取得します。コネクター・フレームワークは Java で記述されており、C++ で記述されたアプリケーション固有のコンポーネントをサポートできるように C++ 拡張が組み込まれています。コネクター・フレームワークのコードはすべてのコネクターで共通です。

コネクター・コントローラー

システムの統合ブローカーが InterChangeServer である場合に使用するコネクター・フレームワークのサブコンポーネント。このサブコンポーネント

は、InterChange Server Express の機能であるコラボレーションと対話します。コネクター・コントローラーは、InterChangeServer 内で動作するもので、アプリケーション固有のビジネス・オブジェクトと汎用ビジネス・オブジェクトの間のマッピングを開始し、コラボレーションのビジネス・オブジェクト定義に対するサブスクリプションを管理します。

統合ブローカー
異種のアプリケーションの間でデータを統合する、WebSphere Business Integration システムのコンポーネント。通常、統合ブローカーはさまざまなサービスを備えています。このサービスには、データをルーティングする機能、統合プロセスを決定する規則のリポジトリー、各種アプリケーションに接続する機能、および統合を容易にする管理機能が含まれます。

WebSphere Business Integration システム
多様なソースの間で情報を移動してビジネス関連の情報を交換し、エンタープライズ環境内の異種のアプリケーションの間で情報の処理とルーティングを行う、エンタープライズ・ソリューション。このビジネス・インテグレーション・システムは、統合ブローカーと 1 つ以上のアダプターで構成されています。

コネクター・アーキテクチャー

図 1 には、コネクター・コンポーネント、コネクター・コンポーネントと WebSphere Business Integration システムとの関係、およびコネクター・コンポーネントの接続先である TCP/IP ネットワークとの関係を示します。

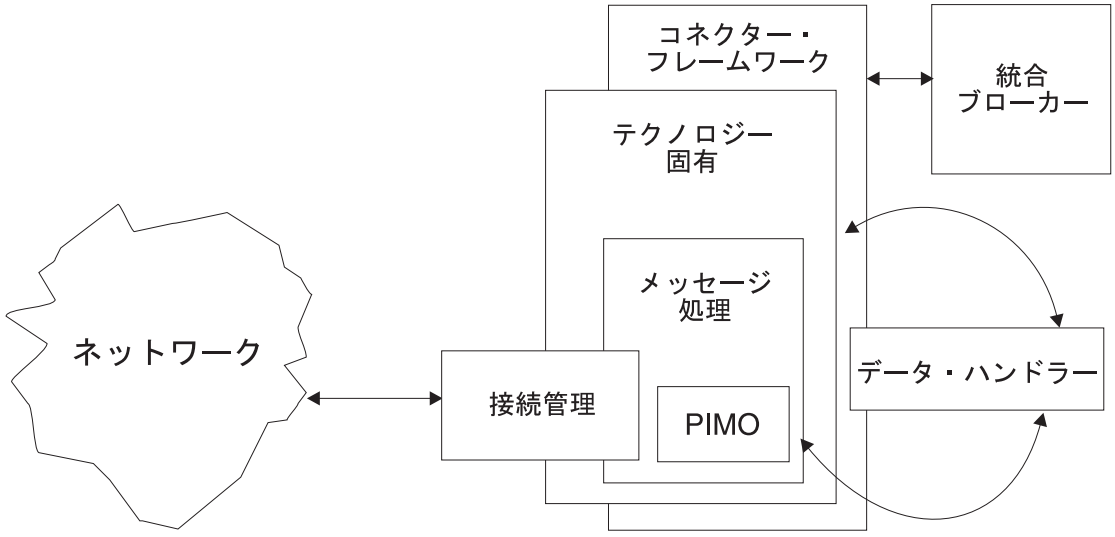

図 1. コネクターのアーキテクチャー

医療業界における HL7 プロトコルなど、業界には固有のメッセージ標準が存在するため、こうした標準を使用すると、データは TCP/IP ネットワークを介して直接送出できます。TCP/IP アダプターは、このようなメッセージを、接続の相手側における TCP アプリケーションの特性に関係なく、WebSphere Business Integration システムの内外に送付するための堅固で拡張が容易な手段を提供します。

イベント処理、つまりインバウンド処理では、TCP アダプターのコネクターは TCP サーバーとして動作し、指定のソケットを listen して、クライアントとの接続が確立されるとソケットにおけるロードのセットアップおよび管理を実行します。データは接続管理コンポーネントからメッセージ処理コンポーネントに転送されます。ここでは、PIMO と呼ばれるサブコンポーネントを使用して、何らかの基本的な前処理が実行されます。前処理が終了すると、コネクターは事前に構成されたデータ・ハンドラーを呼び出して、メッセージを対応するビジネス・オブジェクトに変換します。次に、コネクター・フレームワークは、このオブジェクトを統合ブローカーに公開します。コネクターのインスタンス 1 つにつき、1 つの TCP サーバーを構成できます。

要求処理、つまりアウトバウンド処理では、統合ブローカーが、事前に構成されたターゲット・ホストまたはアプリケーションに送信するメッセージを表すビジネス・オブジェクトをコネクターに送信します。データ・ハンドラーはオブジェクトをメッセージに変換します。このメッセージは、必要に応じて、PIMO サブコンポーネントによる後処理のために送信することもできます。次に、TCP クライアントとして機能するコネクターは、適切なサーバーと通信し、接続を確立して、データの送出を管理します。コネクターは、必要に応じて、ターゲット・サーバーからの応答を処理することもできます。コネクターのインスタンス 1 つにつき、複数の TCP クライアントを構成できます。

第 2 章 コネクターのインストール

この章では、TCP/IP コネクターをインストールするために完了する必要がある作業について説明します。

アダプターの環境

アダプターをインストールするには、まず、次の節に説明されている環境要件を理解する必要があります。

ブローカーとの互換性

アダプターが使用するアダプター・フレームワークは、アダプターと通信する統合ブローカーのバージョンとの互換性を備えている必要があります。Adapter for TCP/IP のバージョン 1.0.x は、以下の統合ブローカーを使用する以下のバージョンのアダプター・フレームワークでサポートされます。

- **アダプター・フレームワーク**: WebSphere Business Integration Adapter Framework バージョン 2.6。このアダプターは、統合ブローカー InterChange Server Express を使用します。

例外については、「*リリース情報*」を参照してください。

前提条件ソフトウェア

Adapter for TCP/IP をインストールおよび構成する前に、以下のソフトウェアをインストールする必要があります。

- すべてのオペレーティング・システム環境で、カスタム・アダプターをコンパイルするために Java コンパイラー (IBM JDK 1.4.2 for Windows 2003) が必要です。
- **Linux:**
  Red Hat Linux AS 3.0 Update 1、Intel (IA32)
  SuSE Linux 8.1、SP3、Intel (IA32)
  SuSE Linux ES 9.0、Intel (IA32)

  注: WebSphere Business Integration Adapter Framework V2.6 の TMTP (Tivoli Monitoring for Transaction Performance) コンポーネントは、Linux Red Hat ではサポートされていません。
- **Windows:**
  WebSphere Business Integration Adapter Framework (管理ツールのみ) の場

合、Windows XP (Service Pack 1A を導入済みのもの)
Windows 2003 (Standard Edition または Enterprise Edition)

– **OS/400 V5R2 および i5/OS V5R3:**
特に明記されていない限り、i5/OS は OS/400 および i5/OS を指します。

ロケール依存データの処理

コネクターは国際化されています。2 バイト文字セット (DBCS) を、同じく 2 バイト文字セットをサポートするインターフェースに送信することができ、また指定された言語のメッセージ・テキストの送信をサポートします。ある文字コードを使用する場所から別の文字コード・セットを使用する場所へデータを転送する場合、コネクターは、そのデータの意味が伝わるように文字変換を実行します。

Java 仮想マシン (JVM) 内部の Java ランタイム環境では、Unicode 文字コード・セットでデータを表現します。Unicode は周知の文字コード・セット (単一バイトとマルチバイトの両方) で文字をエンコードします。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。

コネクターのインストール

このセクションでは、コネクターおよび関連ビジネス・オブジェクトをインストールするために必要な操作を説明します。ソフトウェアの前提条件および互換性については、5 ページの『アダプターの環境』を参照してください。

マシン上で、前提条件となるソフトウェアすべてのインストールを完了すると、コネクターおよびビジネス・オブジェクトをインストールできるようになります。

アダプター (コネクターおよび関連ファイルが格納されている) をインストールする手順の詳細については、次の Web サイトにある「*WebSphere Business Integration Server Express インストール・ガイド (Windows 版)*」、「*WebSphere Business Integration Server Express インストール・ガイド (Linux 版)*」、または「*WebSphere Business Integration Server Express インストール・ガイド (OS/400 版)*」を参照してください。

http://www.ibm.com/websphere/wbiserverexpress/infocenter

インストールの検証

コネクターをインストールすると、次の表のファイルがマシンにインストールされます。表中の *ProductDir* は、IBM WebSphere Business Integration Server Express アダプター・ソフトウェアをインストールしたディレクトリーを指します。

インストールされる Windows ファイル構造

以下の表に、Windows マシンにインストールする場合にコネクターで使用されるファイル構造を示します。

表 2. コネクターとともにインストールされるファイル - Windows

ディレクトリー	インストールされるファイル
ProductDir¥bin¥Data¥App	BIA_TCPIPAdapterTemplate: コネクターの構成ファイル・テンプレート
ProductDir¥connectors¥TCPIP	BIA_TCPIP.jar: 主なコネクター・コード start_TCPIP.bat: コネクター始動バッチ・ファイル
ProductDir¥connectors¥messages¥ BIA_TCPIPadapter.txt	BIA_TCPIPadapter.txt: コネクター・メッセージ・ファイル。エラー・メッセージおよびコードが格納されている。
ProductDir¥repository¥ TCPIP¥Sample	• コネクターが内部処理のために使用するビジネス・オブジェクトおよびメタオブジェクトの定義ファイル • TCP/IP コネクターによって構成されたサンプル・アプリケーション。NCPDP、HL7 over LLP、text/name-value と組み合わせて使用するためのサンプルを含む。
ProductDir¥connectors¥TCPIP¥ *start_TCPIP_service.bat*	コネクター・サービスのサービス始動スクリプト。アダプターをサービスとして開始します。このスクリプトは、すべてのアダプターをサービスとして開始および停止できるサービス制御パネルから呼び出されます。
ProductDir¥connectors¥TCPIP ¥start_tcpip.bat	通常の始動スクリプト。アダプターを通常のプロセスとして開始します。

インストールされる Linux ファイル構造

以下の表に、Linux マシンにインストールする場合にコネクターで使用されるファイル構造を示します。

表 3. コネクターとともにインストールされるファイル - Linux

ディレクトリー	インストールされるファイル
ProductDir/bin/Data/App	BIA_TCPIPAdapterTemplate: コネクターの構成ファイル・テンプレート
ProductDir/connectors/TCPIP	BIA_TCPIP.jar: 主なコネクター・コード start_TCPIP.sh: コネクター始動シェル・ファイル
ProductDir/connectors/ messages/ BIA_TCPIPAdapter.txt	BIA_TCPIPAdapter.txt: コネクター・メッセージ・ファイル。エラー・メッセージおよびコードが格納されている。
ProductDir/repository/TCPIP/Sample	• コネクターが内部処理のために使用するビジネス・オブジェクトおよびメタオブジェクトの定義ファイル • TCP/IP コネクターによって構成されたサンプル・アプリケーション。NCPDP、HL7 over LLP、text/name-value と組み合わせて使用するためのサンプルを含む。

インストールされる i5/OS ファイル構造

以下の表に、i5/OS マシンにインストールする場合にコネクターで使用されるファイル構造を示します。

表 4. コネクターとともにインストールされるファイル - i5/OS

ディレクトリー	インストールされるファイル
ProductDir/connectors/TCPIP	BIA_TCPIP.jar: 主なコネクター・コード start_TCPIP.sh: コネクター始動シェル・ファイル
ProductDir/connectors/messages/ BIA_TCPIPAdapter.txt	BIA_TCPIPAdapter.txt: コネクター・メッセージ・ファイル。エラー・メッセージおよびコードが格納されている。
ProductDir/repository/TCPIP/Sample	• コネクターが内部処理のために使用するビジネス・オブジェクトおよびメタオブジェクトの定義ファイル • TCP/IP コネクターによって構成されたサンプル・アプリケーション。NCPDP、HL7 over LLP、text/name-value と組み合わせて使用するためのサンプルを含む。

第 3 章 TCP/IP アダプターのビジネス・オブジェクトおよびメタオブジェクト

Adapter for TCP/IP のランタイム・コンポーネントであるコネクターは、医療業界の HL7 プロトコルなど、既知のプロトコルの下位に位置する TCP/IP ネットワークを介して、WebSphere Business Integration システムの内外に直接送信されるデータを転送するための汎用のコンジットとして設計されています。インバウンド情報、つまりイベント情報は、コネクターによってネットワーク・メッセージ・ストリームから収集され、WBI ビジネス・オブジェクトに変換されて、統合ブローカーに公開されます。アウトバウンド情報、つまりサービス呼び出し要求は、WBI ビジネス・オブジェクトとして統合ブローカーから受信され、ネットワーク・メッセージ・ストリームに変換されて、ネットワークを介して返送されます。このフローでの WBI ビジネス・オブジェクトの性質は、コネクター構成ファイルの設定に基づいてコネクターが呼び出すデータ変換プラグインであるデータ・ハンドラーに全面的に依存します。アダプターそのものではなく、データ・ハンドラーは、適切な WBI ビジネス・オブジェクト形式との間でメッセージを変換します。

ただし、データ管理に役立ち、コネクター内部で処理の流れを誘導するその他のオブジェクトが、次の 3 種類存在します。

- 『内部ビジネス・オブジェクト』
- 11 ページの『汎用メタオブジェクト』
- 15 ページの『PIMO (Production Instruction Meta Object) フレームワーク・メタオブジェクト』

次のセクションでは、これらのオブジェクト、コネクターの内部処理におけるこれらのオブジェクトの機能、およびこれらのオブジェクトの構造について説明します。

内部ビジネス・オブジェクト

コネクターの内部ビジネス・オブジェクトは、(イベント・モード時に) データをネットワークから取り出すときや、(サービス呼び出し要求モード時に) データをネットワークに送り出すときに、過渡的なデータ・ラッパーとして使用されます。

BIA_ContentBO

イベント・モードでは、コネクターは TCP サーバーとして動作し、リモート・アプリケーションからの要求をソケットで listen して、データを送信するチャネルを確立します。接続管理サブコンポーネントは、接続を確立し、ネットワークからの着信データのストリームを管理します。この内容としては、ロード・バランシングや、複数の要求を処理するための並列処理の設定などがあります。データは、内部に流れてくると、メッセージ処理コンポーネントに渡されます。ここでデータは、基本的なデータ・ラッパーである BIA_ContentBO に保持されます。

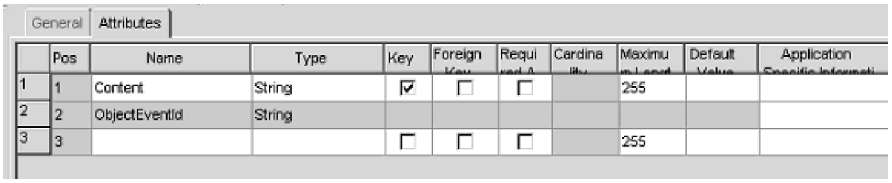

図 2. Business Object Designer Express での BIA_ContentBO

BIA_ContentBO に該当する属性は次のとおりです。

BO レベルの属性	説明
Content	アプリケーション・データまたはプロトコル・データを格納します。

BIA_InputMessage

Content オブジェクトは、BIA_InputMessage ビジネス・オブジェクトの内部に格納されています。入力メッセージ・オブジェクトには、最初、リモート・アプリケーションからの完成したメッセージと未完成のメッセージが格納されています。コネクターは、完成したメッセージと未完成のメッセージを分離し、未完成のメッセージは完成するまでキューに入れます。完成したメッセージは BIA_InputMessage オブジェクトでラップして、PIMO フレームワークに送信します。ここでは、メッセージに対して何らかの形の前処理を実行してから、最終処理のためにデータ・ハンドラーに渡します。

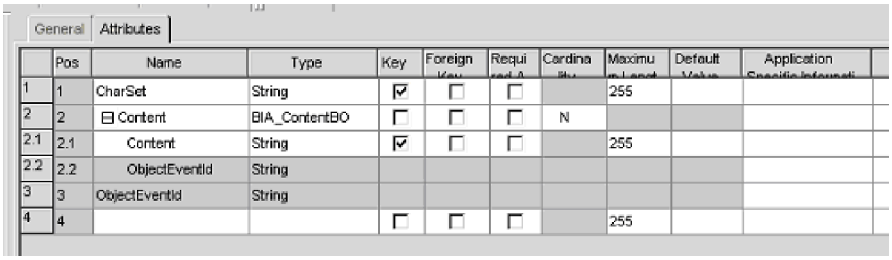

図 3. Business Object Designer Express での BIA_InputMessage

BIA_InputMessage オブジェクトに該当する属性は次のとおりです。

BO レベルの属性	説明
CharSet	着信バイトに適用されるエンコード方式
Content	ソケットに到着する内容を BIA_ContentBO として格納します。属性の長さは着信メッセージの長さと比較して制限されるため、この属性のカーディナリティーは N になります。

BIA_ApplicationMessage

サービス呼び出し要求モードでは、リモート・アプリケーションに送信されるメッセージを表す WBI ビジネス・オブジェクトは、統合ブローカーから送信されま

す。これらのオブジェクトは、データ・ハンドラーによって適切なメッセージ形式に変換されます。コネクターは、このメッセージ・データを、BIA_ApplicationMessage オブジェクトに格納されている BIA_ContentBO でラップします。このメッセージ・データには、PIMO 後処理が実行される場合があります。そのタイミングは、TCP クライアントとして動作するコネクターが、TCP/IP ネットワークを介して、コネクター構成ファイルで指定されている宛先にメッセージ・データを返送した後になります。

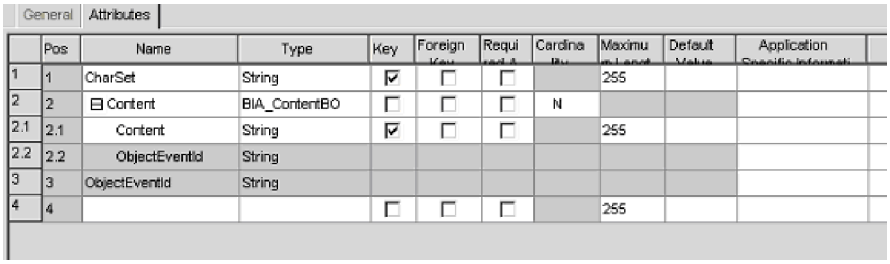

図 4. Business Object Designer Express での BIA_ApplicationMessage

BIA_ApplicationMessage オブジェクトに該当する属性は次のとおりです。

BO レベルの属性	**説明**
Charset	着信バイトに適用されるエンコード方式
Content	データ・ハンドラーから到着する内容を BIA_ContentBO として格納します。属性の長さは着信メッセージの長さと比較して制限されるため、この属性のカーディナリティーは N になります。

コネクターが使用する汎用の内部ビジネス・オブジェクトは、その他に 2 つあります。1 つは、BIA_ResponseMessage オブジェクトで、ここには、サービス呼び出し要求の結果として、リモート・アプリケーションからの確認通知メッセージが (Content オブジェクトにラップされた状態で) 格納されています。もう 1 つは BIA_FinalMessage オブジェクトで、ここには、接続管理サブコンポーネント自体の情報が格納されています。すべての内部ビジネス・オブジェクトおよびメタオブジェクトの定義ファイルは、次のディレクトリーに XML スキーマ・ファイル (.xsd) として格納されています。Windows の場合は *ProductDir*¥connectors¥TCPIP¥Samples で、Linux の場合は *ProductDir*/connectors/TCPIP/Samples です。これらを表示するには、Business Object Designer Express または XML 対応のブラウザーを使用します。

汎用メタオブジェクト

コネクター構成ファイルの 3 つのプロパティーに対応し、これらのプロパティーによって設定される汎用メタオブジェクトのグループが、コネクター内に 3 つ存在します。これらは、構成メタオブジェクト、サービス登録メタオブジェクト (実際にはネストされた一連のオブジェクト)、およびデータ・ハンドラー・メタオブジェクト (これもネストされた一連のオブジェクト) です。

構成メタオブジェクト

構成メタオブジェクトとは、BIA_Static_MO のことです。このメタオブジェクトには、さまざまなメッセージ・タイプを処理するための静的なメタ情報がアプリケーション固有の情報 (ASI) の値として格納されます。

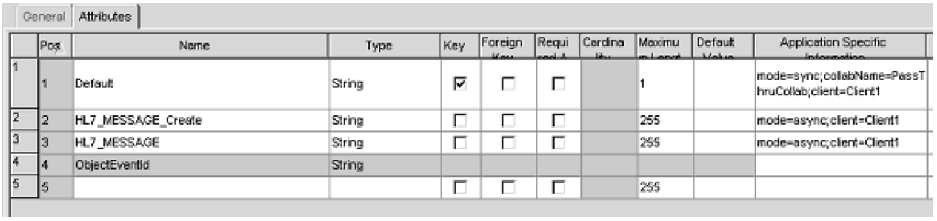

図 5. Business Object Designer Express での BIA_Static_MO

この例のオブジェクトに該当する属性は次のとおりです。

MO レベルの属性	**説明**
Default	デフォルト値が記述されます。
HL7_Message_Create	HL7_Message オブジェクトに関連した Create 動詞の静的メタ属性値が記述されます。
HL7_Message	HL7_Message オブジェクトに関連した静的メタ属性値が記述されます。

指定したメッセージ・タイプごとに ASI の値として格納されているメッセージ・タイプを処理するための静的メタ情報の種類は、次のとおりです。

メッセージ・タイプごとのアプリケーション固有の情報	**説明**
mode =	可能な値は sync または async です。「sync」の場合、コネクターはイベント処理時に同期方式でブローカーと通信します。「async」の場合、通信は非同期になります。
collabName =	同期イベント処理の場合にのみ関係があります。コネクターが呼び出す必要があるコラボレーションに名前を付けます。
client = Clientx	サービス呼び出し要求処理の場合にのみ関係があります。コネクターの構成属性 Clientx に格納されているリモート・サーバー構成情報に名前を付けます。

サービス登録メタオブジェクト

サービス登録メタオブジェクトとは、BIA_MO_Service のことです。これは、メッセージ・タイプの処理時に使用できる複数の種類の「サービス」を記述する一連のオブジェクトのうち、トップレベルのオブジェクトです。「サービス」とは、再使用可能な任意の機能のことです。このリリースでは「データ・ハンドラー・サービ

ス」のみを定義しますが、データベース・アクセス用の「データベース・サービス」など、その他のサービスも定義できます。

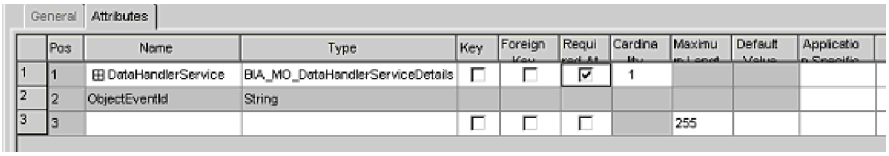

図 6. Business Object Designer Express での BIA_MO_Service

このオブジェクトに該当する属性は次のとおりです。

MO レベルの属性	**説明**
DataHandlerService	詳細な情報が格納されている BIA_MO_DataHandlerServiceDetails を参照します。

BIA_MO_DataHandlerServiceDetails オブジェクトは、データ・ハンドラー・サービスの追加情報を提供します。

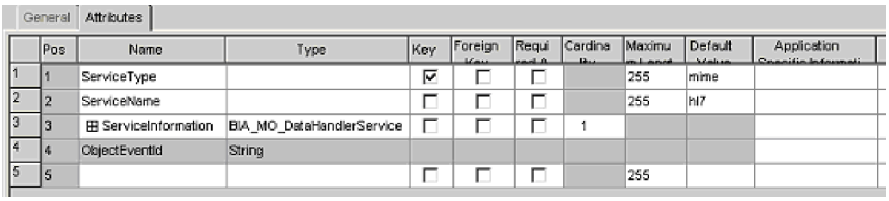

図 7. Business Object Designer Express での BIA_MO_DataHandlerServiceDetails

このオブジェクトに該当する属性は次のとおりです。

MO レベルの属性	**説明**
Service Type	サービス・タイプを指定するための追加情報が格納されます。コネクターの処理のためには使用されません。
ServiceName	サービス名を指定します。DataHandlerService の場合、これは、DataHandlerMimeType 属性のコネクター構成ファイルに指定されているように、処理の対象となるメッセージの MIME タイプになります。
ServiceInformation	詳細な情報が格納されている BIA_MO_DataHandlerService を参照します。

BIA_MO_DataHandlerService オブジェクトは、適切なデータ・ハンドラーおよびそのメソッドを呼び出すための情報を提供します。

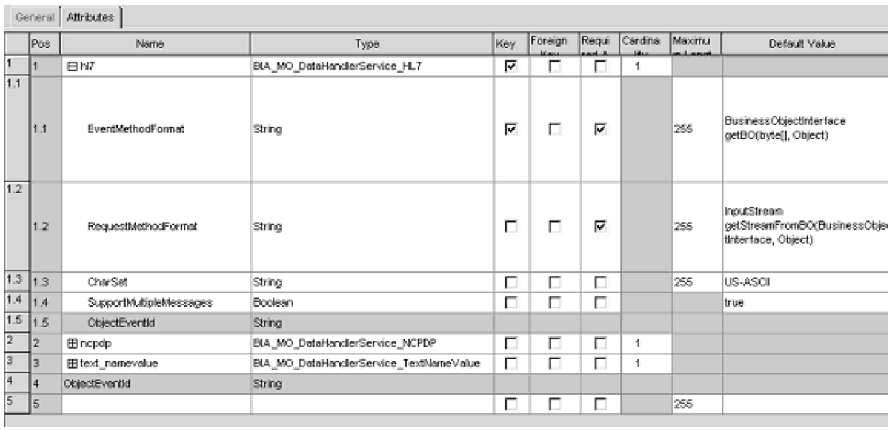

図 8. Business Object Designer Express での BIA_MO_DataHandlerService

このオブジェクトに該当する属性は次のとおりです。

MO レベルの属性	**説明**
hl7	特定の MIME タイプの処理を定義します。この属性で参照されるオブジェクト (例は BIA_MO_DataHandlerService_HL7) には、データ・ハンドラーが使用する特定の情報が格納されています。後述する関連の子属性 (+ 記号で表示) は、格納される側のオブジェクトに属します。これらの子属性は、ここに示す MIME タイプが BIA_MO_DataHandlerServiceDetails オブジェクトの ServiceName 属性に一致した場合に使用されます。
+EventMethodFormat	イベント処理時に呼び出される HL7 DataHandler メソッドを格納します。
+RequestMethodFormat	サービス呼び出し要求処理時に呼び出される HL7 DataHandler メソッドを格納します。
+CharSet	ソケットに到着するメッセージ・データの CharSet を格納します。
+SupportMultipleMessages	到着するデータに含まれるメッセージが 1 つか複数かを示します。
ncpdp	別のタイプ固有オブジェクト (ここでの例は BIA_MO_DataHandlerService_NCPDP) を参照することにより、2 番目の MIME タイプの処理を定義します。

データ・ハンドラー・メタオブジェクト

データ・ハンドラー・メタオブジェクトとは、BIA_MO_DataHandler_Default のことです。これは、指定のデータ・ハンドラーが使用する情報が格納されている階層で、トップレベルのオブジェクトです。この情報は、サービス登録オブジェクト階層に格納されている情報とは異なります。

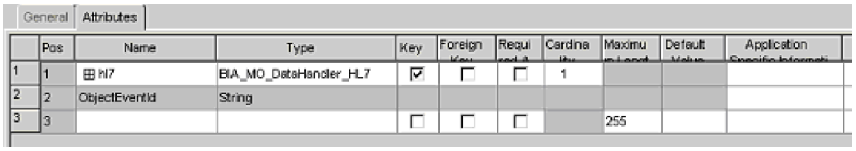

図 9. Business Object Designer Express での BIA_MO_DataHandlerDefault

このオブジェクトに該当する属性は次のとおりです。

MO レベルの属性	説明
hl7	DataHandlerMimeType 属性のコネクター構成ファイルに指定したとおりに MIME タイプの処理を定義します。参照されるオブジェクト (この例では BIA_MO_DataHandler_HL7) には、データ・ハンドラーが使用する、タイプ固有の情報が格納されています。

PIMO (Production Instruction Meta Object) フレームワーク・メタオブジェクト

Production Instruction Meta Object フレームワークは、コネクター内部での特定の種類のオブジェクト操作を実行するための抽象的な手段を提供します。TCP/IP のアダプターの場合、この手段を使用すると、メッセージ・データの前処理および後処理が実行できます。この手段そのものの詳細は、75 ページの『付録 C. アダプターの処理』で説明しますが、このフレームワークが使用する主なメタオブジェクトは、情報をすべて記載するためにここで説明します。

BIA_MO_Tcpip_MapSubscriptions

PIMO の処理は、1 組のマッピング・メタオブジェクトによって配列された一連の変換が基本になります。マップ階層の最上位は、BIA_MO_Tcpip_MapSubscriptions オブジェクトです。

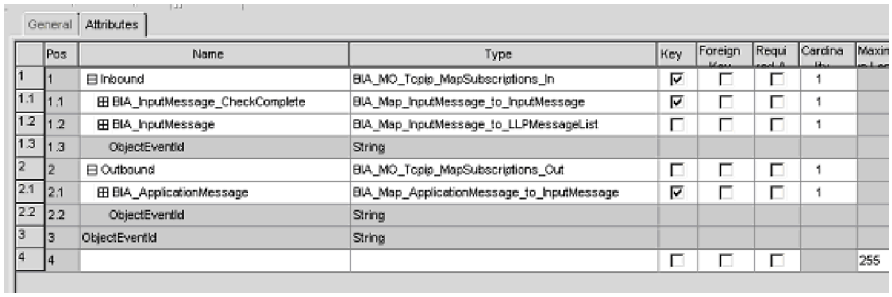

図 10. Business Object Designer Express での BIA_MO_Tcpip_MapSubscriptions

このオブジェクトに該当する属性は次のとおりです。

MO レベルの属性	説明
Inbound	イベント・モード処理で PIMO が使用するマップの組を示します。この例では 2 つのマップを使用します。
+BIA_InputMessage_CheckComplete	子属性。複数のオブジェクトを分離するために使用するマッピング・メタオブジェクト (BIA_Map_InputMessage_to_InputMessage) が格納されます。BIA_MO_DataHandlerService オブジェクトの SupportMultipleMessages 値は true に設定されていることが前提です。
+BIA_InputMessage	子属性。メインの HL7 データから LLP ヘッダーおよびトレーラーを削除するときに使用するマッピング・メタオブジェクト (BIA_Map_InputMessage_to_LLPMessageList) が格納されます。 詳細については、後述のセクションを参照してください。
Outbound	サービス呼び出し要求モード処理で PIMO が使用するマップの組を示します。この例では 1 つのマップを使用します。
+BIA_ApplicationMessage	子属性。マッピング・メタオブジェクト (BIA_Map_ApplicationMessage_to_InputMessage) が格納されます。

BIA_Map_InputMessage_to_LLPMessageList

BIA_Map_InputMessage_to_LLPMessageList マップ・オブジェクトには、前述のマップ・サブスクリプション・メタオブジェクトの例で示したインバウンド・マッピング処理の 2 番目の段階を実行するために PIMO が必要とする手順が記録されています。

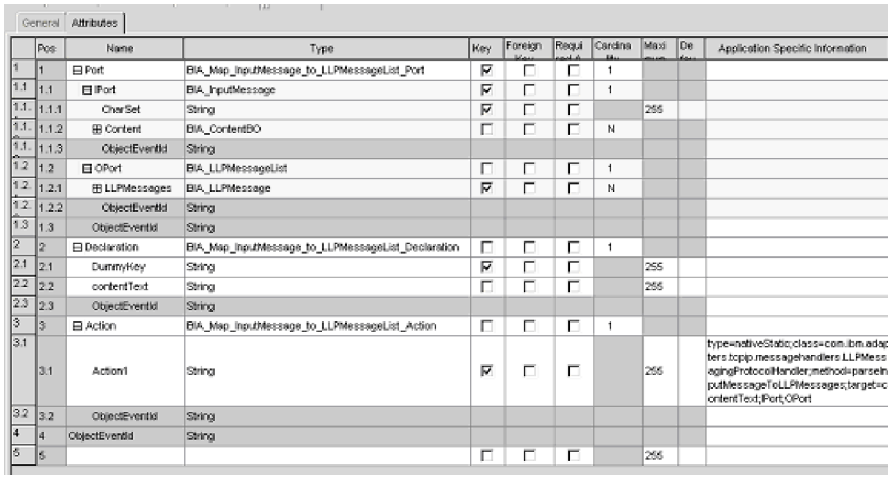

図 11. Business Object Designer Express での BIA_Map_InputMessage_to_LLPMessageList

このオブジェクトに該当する属性は次のとおりです。

マップ BO レベルの属性	説明
Port	各 PIMO マップ・オブジェクトには、IPort と OPort の 2 つのポートがあります。これらは、BIA_InputMessage_to_LLPMessageList_Port の例で定義されています。 これらは、送信元オブジェクト (この例では BIA_InputMessage) と宛先オブジェクト (この例では BIA_LLPMessageList) の予想タイプを示しています。
Declaration	PIMO マップ・オブジェクトには、処理中に使用する一時変数の名前が書き込まれている Declaration オブジェクトが組み込まれている場合があります。この例では、Declaration オブジェクト (BIA_Map_InputMessage_to_LLPMessageList_Declaration) に、「contentText」という名前が入っています。
Action	各 PIMO マップ・オブジェクトは、処理を実行する実際のクラスとメソッドを指す情報が格納されている Action オブジェクト (ここでは BIA_Map_InputMessage_to_LLPMessageList) を 1 つ以上備えています。

PIMO マップ・オブジェクトに格納されている Action オブジェクトによって定義されるアクションごとの ASI 値は、次のとおりです。

定義されたアクションごとのアプリケーション固有の情報	説明
type =″nativeStatic″	呼び出される Java クラスの静的メソッドを示します。現在、PIMO がサポートしているのは、静的メソッドのみです。
class =	Java クラスの完全修飾名です。この例の場合、クラスは `com.ibm.adapters.tcpip.messagehandlers` パッケージの `LLPMessagingProtocolHandler` クラスです。
method =	呼び出されるメソッドです。この例では、メソッドは `parseInputMessageToLLPMessages` メソッドです。
target =	戻されたデータの格納場所です。この例では、データの格納先は Declaration 属性に作成された変数 `contentText` です。

定義されたアクションごとの アプリケーション固有の情報	説明
var1, var2 ... varx	メソッドに渡されるパラメーターのオープン・リストです。このリストにおける変数の順序と数は、メソッドが予想するパラメーターの順序と数に完全に一致する必要があります。この例では、var1 と var2 は、それぞれ Port 属性の IPort および OPort になります。

第 4 章 コネクターの構成

Connector Configurator Express とは、Adapter for TCP/IP に付属のツールで、このツールを使用すると、付属のコネクター・テンプレートを構成できます。

Connector Configurator Express の概要

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

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

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成する。
- **コネクター構成ファイル**を作成します。インストールするコネクターごとに構成ファイルを 1 つ作成する必要があります。
- 構成ファイル内のプロパティーを設定する。
  場合によっては、コネクター・テンプレートでプロパティーに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、InterChange Server Express 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 Express 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。使用している統合ブローカーが InterChange Server Express であるため、Connector Configurator Express を System Manager から実行します (20 ページの『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で使用されるので、最初から定義する必要はありません。構成ファイルを作成すると、Connector Configurator Express によって標準プロパティーがそのファイルに挿入されます。ただし、Connector Configurator Express で各標準プロパティーの値を設定する必要があります。

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

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

ンプレートが作成されていない場合には、21 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

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

Connector Configurator Express の始動

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

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

スタンドアロン・モードでの Configurator の実行

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

これを行うには、以下のステップを実行します。

- 「**スタート**」>「**すべてのプログラム**」から、「**IBM WebSphere Business Integration Express**」>「**Toolset Express**」>「**開発**」>「**Connector Configurator Express**」をクリックします。
- 「**ファイル**」>「**新規**」>「**コネクター構成**」を選択します。
- 「**システム接続: Integration Broker**」の隣のプルダウン・メニューをクリックします。これで InterChange Server Express を選択できます。

Connector Configurator Express を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存する方法が便利です (26 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 Express を実行できます。

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

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

コネクター固有のプロパティー・テンプレートの作成

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

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

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

新規テンプレートの作成

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

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

1. 「**ファイル**」>「**新規**」>「**コネクター固有プロパティー・テンプレート**」をクリックします。
2. 「**コネクター固有プロパティー・テンプレート**」ダイアログ・ボックスが表示されます。
   - 「**新規テンプレート名を入力してください**」の下にある「**名前**」フィールドに、新規テンプレートの名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
   - テンプレートに含まれているコネクター固有のプロパティー定義を調べるには、「**テンプレート名**」表示でそのテンプレートの名前を選択します。そのテンプレートに含まれているプロパティー定義のリストが「**テンプレートのプレビュー**」表示に表示されます。
3. テンプレートを作成するときには、ご使用のコネクターに必要なプロパティー定義に類似したプロパティー定義が含まれている既存のテンプレートを使用できま

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

- 既存のテンプレートを変更する予定の場合は、「**変更する既存のテンプレートを選択してください: 検索テンプレート**」の下にある「**テンプレート名**」テーブルのリストからテンプレートの名前を選択します。
- このテーブルには、現在使用可能なすべてのテンプレートが表示されます。テンプレートを検索することもできます。

一般特性の指定

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

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

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

値の指定

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

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

新規のプロパティー値は、以下のように作成します。

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

「**値**」パネルには、次のような 3 つの列があるテーブルが表示されます。

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

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

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

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

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

依存関係の設定

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

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

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

   == (等しい)

   != (等しくない)

   > (より大)

   < (より小)

   >= (より大か等しい)

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

新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、構成ファイルの名前を指定し、統合ブローカーを選択する必要があります。

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

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

- 「**Integration Broker**」フィールドで、InterChange Server Express を選択します。
- この章で後述する説明に従って、「**新規コネクター**」ウィンドウの残りのフィールドに入力します。

コネクター固有のテンプレートからの構成ファイルの作成

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

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

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

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

     「InterChange Server Express」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 Express のメッセージ・パネルの状況表示に、構成ファイルが正常に作成されたことが示されます。

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

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

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

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

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

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

     InterChange Server Express 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 Express を始動します。
3. 「**ファイル**」>「**開く**」>「**プロジェクトから**」をクリックします。

構成ファイルの完成

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

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。InterChange Server Express がブローカーであ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 InterChange Server Express を選択します。
2. 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 29 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 Express 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 Express 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、または既存のコネクター構成ファイルを開くと、Connector Configurator Express に構成画面が表示されます。構成画面には、必要な構成値のカテゴリーに対応する複数のタブがあります。

Connector Configurator Express では、InterChange Server Express で実行されるコネクターで、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。

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

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

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

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

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

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下の手順を実行します。

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

アプリケーション固有の構成プロパティーの設定

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

1. グリッドの左上端の部分で右マウス・ボタンをクリックします。ポップアップ・メニュー・バーが表示されます。プロパティーを追加するときは「**追加**」をクリックします。子プロパティーを追加するには、親の行番号で右マウス・ボタンをクリックし、「**子を追加**」をクリックします。
2. プロパティーまたは子プロパティーの値を入力します。
3. プロパティーを暗号化するには、「**暗号化**」ボックスを選択します。
4. 28 ページの『標準コネクター・プロパティーの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

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

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

コネクター・プロパティーの暗号化

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

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

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

更新メソッド

付録 A 『*標準構成プロパティー*』の 46 ページの『構成プロパティー値の概要』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 Express の「**サポートされているビジネス・オブジェクト**」タブを使用します。汎用ビジネス・オブジェクトと、アプリケーション固有のビジネス・オブ

ジェクトの両方を指定する必要があり、またそれらのビジネス・オブジェクト間のマップの関連を指定することが必要です。

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。

ご使用のブローカーが InterChange Server Express である場合

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

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

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

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

1. ビジネス・オブジェクト・フィールドを選択するため、そのビジネス・オブジェクトの左側の番号をクリックします。
2. 「Connector Configurator Express」ウィンドウの「**編集**」メニューから、「**行を削除**」をクリックします。リスト表示からビジネス・オブジェクトが除去されます。
3. 「**ファイル**」メニューから、「**プロジェクトの保管**」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 Express」ウィンドウでは、「エージェント・サポート」を選択しても問題ないかどうかの検証は行われません。

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

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

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

関係付けられたマップ (InterChange Server Express のみ)

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

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

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

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

「**関連付けられたマップ**」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的**

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

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

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

  以下の手順を実行して、マップを明示的にバインドします。

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

トレース/ログ・ファイル値の設定

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

ログとトレースの値を変更するには、以下の手順を実行します。

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

     注: STDOUT オプションは、Windows プラットフォームで実行しているコネクターの「**トレース/ログ・ファイル**」タブでのみ使用できます。
     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。

- ファイルに: (i5/OS プラットフォームはこのオプションのみを使用します。)

  注: i5/OS プラットフォームでは、エージェントが iSeries で稼働している場合、iSeries ファイル・システム・パスを手動で指定する必要があります。iSeries のマップされたドライブを検出するために「**参照**」ボタンを使用しないでください。例えば、パスを /QIBM/UserData/WBIServer44/servername/log/adapter.log のように手動で指定します。

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『標準構成プロパティー』にある `ContainerManagedEvents` の下の説明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 Express 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 Express のタイトル・バーには、現在のブローカー・モード (InterChange Server Express) が常に表示されます。

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

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

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、次のガイドを参照してください。

- InterChange Server Express の場合: 「*システム・インプリメンテーション・ガイド*」

構成の完了

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

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

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

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

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

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

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

例えば、Locale プロパティーの値のリストにロケール en_GB を追加するには、stdConnProps.xml ファイルを開き、以下に太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>
```

第 5 章 コネクターの実行

この章では、コネクターの始動と停止のために必要な手順、および同じマシン上でコネクターの複数インスタンスを実行するために必要な手順について説明します。この章は、以下のセクションから構成されています。

- 『コネクターの始動』
- 37 ページの『コネクターの停止』
- 38 ページの『1 つのサーバー上での接続の複数インスタンスの作成』

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。Windows システムでは、始動スクリプトは、コネクターのランタイム・ディレクトリー *ProductDir*¥connectors¥*tcpip* に存在していなければなりません。

Linux システムでは、始動スクリプトは、*ProductDir*/bin ディレクトリーに存在していなければなりません。

i5/OS システムでは、始動スクリプトは、コネクターの実行に使用される /QIBM/UserData/WBIServer44/<instance>/connectors/<ConnInstance>/ に存在していなければなりません。

始動スクリプトの名前は、表 5 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 5. 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
Linux	connector_manager
i5/OS	start_tcpip.sh
Windows	start_*tcpip*.bat

始動スクリプトが実行されると、始動スクリプトはデフォルトでは *Productdir* に構成ファイルがあるものと考えます (下記のコマンドを参照)。ここに構成ファイルを配置します。

注: アダプターが JMS トランスポートを使用している場合、ローカル構成ファイルが必要です。

- **Windows システムでのコネクターの始動:**
  - 「**スタート**」メニューから、「**プログラム**」>「**IBM WebSphere Business Integration Express**」>「**アダプター**」>「**コネクター**」を選択します。デフォルトでは、プログラム名は「IBM WebSphere Business Integration Server Express」となっていますが、カスタマイズ可能です。あるいは、ご使用のコネクターへのデスクトップ・ショートカットを作成することもできます。
  - Windows コマンド行から start_connName connName brokerName {-cconfigFile} を実行します。

– 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。

- **Linux システムでのコネクターの始動**:
  – コマンド行から次のように入力します。

    ```
    connector_manager -start connName brokerName [-cconfigFile ]
    ```

    ここで、*connName* はコネクターの名前であり、*brokerName* は統合ブローカーを示します。
  – InterChange Server Express の場合は、*brokerName* に InterChange Server Express インスタンスの名前を指定します。
- **i5/OS システムでのコネクターの始動**:
  – WebSphere Business Integrations Server Express コンソールがインストールされている Windows システムから、「**IBM WebSphere Business Integration Server Express**」>「**Toolset Express**」> 「**管理**」>「**コンソール**」を選択します。次に、i5/OS システム名または IP アドレス、および *JOBCTL 特殊権限を持つユーザー・プロファイルおよびパスワードを指定します。コネクターのリストからコネクターを選択し、「開始」をクリックします。
  – コンソールを使用して自動的にアダプターを開始するには、submit_adapter.sh script を使用します。アダプター・エージェントを、自動的に開始するようにコンソールから設定することもできます。このオプションを設定するには、アダプター・エージェントを右クリックし、「**自動始動**」 チェック・ボックスを使用可能になるようにクリックします。
  – バッチ・モードでは、i5/OS コマンド行から、CL コマンド QSH を実行する必要があります。また、QSHELL 環境から、
    /QIBM/ProdData/WBIServer44/bin/submit_adapter.sh *connName WebSphereICSName pathToConnNameStartScript jobDescriptionName* を実行します。ここで、*connName* はコネクター名、*WebSphereICSName* は Interchange Server Express サーバー名 (デフォルトは QWBIDFT44)、*pathToConnNameStartScript* はコネクター始動スクリプトへの絶対パス、*jobDescriptionName* は QWBISVR44 ライブラリーで使用するジョブ記述の名前です。
  – 対話モードで、
    connector_manager -start connName WebSphereICSName [-cconfigFile] と指定します。InterChange Server Express の場合は、WebSphereICSName に InterChange Server Express インスタンスの名前を指定します。

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、「*システム管理ガイド*」を参照してください。

コネクターの始動時に実行されるタスク

コネクターを始動すると、以下のタスクが実行されます。

- 構成情報の検索
- サポートされている内部ビジネス・オブジェクト定義の検索
- コネクター・バージョンのリターン

- 内部ビジネス・オブジェクト・ハンドラーを指すポインターのリターン
- データ・ハンドラーを指すポインターの検索

コネクターの停止

コネクターを停止する方法は、コネクターが始動された方法によって異なります。

- **Windows**:
  - 始動スクリプトを起動すると、そのコネクター用の個別の「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。
  - 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。
- **i5/OS**:
  - コンソールを使用して、または QSHELL で「submit_adapter.sh」スクリプトを使用してコネクターを始動した場合、以下の 2 つの方法のいずれかを使用してコネクターを停止することができます。
  - WebSphere Business Integration Server Express コンソールがインストールされている Windows システムから、「**IBM WebSphere Business Integration Server Express**」>「**Toolset Express**」>「**管理**」>「**コンソール**」を選択します。次に、i5/OS システム名または IP アドレス、および *JOBCTL 特殊権限を持つユーザー・プロファイルおよびパスワードを指定します。リストから iSeries アダプターを選択し、「停止」ボタンを選択します。 CL コマンド WRKACTJOB SBS (QWBISVR44) を使用して、Server Express 製品へのジョブを表示します。リストをスクロールし、コネクターのジョブ記述に一致するジョブ名のジョブを検出します。CL コマンド WRKSMBJOB を実行してジョブを表示します。ジョブ名は QWBIISRSC です。このジョブに対してオプション 4 を選択し、F4 を押して ENDJOB コマンドのプロンプトを取得します。次に、オプション・パラメーターに *IMMED を指定し、Enter を押します。

    注: QWBISVR44 サブシステムが終了すると、コネクターは終了します。
  - start_connName.sh スクリプトを使用して QSHELL からアダプターを始動した場合は、F3 キーを押してコネクターを終了します。/QIBM/ProdData/WBIServer44/bin ディレクトリーにある *stop_adapter.sh*という名前のスクリプトを使用して、エージェントを停止することもできます。
- **Linux**:

  コネクターはバックグラウンドで実行されるため、個別のウィンドウを使用しません。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 -stop connName
  ```

  ここで、*connName* はコネクターの名前です。

1 つのサーバー上での接続の複数インスタンスの作成

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

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

新規ディレクトリーの作成

- **Windows プラットフォームの場合**:

  ProductDir¥connectors¥connectorInstance

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

  ProductDir¥repository¥connectorInstance

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

  InterChange Server Express サーバー名を startup.bat のパラメーターとして指定することができます。例えば、start_iseries.bat connName serverName とします。

- **i5/OS プラットフォームの場合:**

  `/QIBM/UserData/WBIServer44/WebShereICSName/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ており、WebSphereICSName はコネクター実行に使用される InterChange Server Express インスタンスの名前です。

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

  /QIBM/UserData/WBIServer44/WebSphereICSName/repository
  /connectorInstance。ここで、WebSphereICSName は、コネクター実行に使用される InterChange Server Express インスタンスの名前です。

- **Linux プラットフォームの場合**:
  ProductDir/connectors/connectorInstance。ここで connectorInstance はコネクター・インスタンスを一意的に示します。コネクターに、コネクター固有のメタオブジェクトがある場合、コネクター・インスタンス用のメタオブジェクトを作成する必要があります。メタオブジェクトをファイルとして保管する場合、ProductDir/repository/connectorInstance ディレクトリーを作成し、ファイルをここに保管します。InterChange Server Express サーバー名を connector_manager のパ

ラメーターとして指定することができます。例えば、connector_manager -start connName WebSphereICSName [-cConfigFile] のようにします。

ビジネス・オブジェクト定義の作成

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

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

以下のステップを使用して、Connector Configurator Express 内で、コネクター・インスタンスの構成ファイル (コネクター定義) を作成します。

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、『ビジネス・オブジェクト定義の作成』で作成したコネクター・ディレクトリーに格納します。
3. (Windows のみ) 始動スクリプトのショートカットを作成します。
4. (Windows のみ) 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。
5. (i5/OS のみ) 以下の情報を使用して、コネクターのジョブ記述を作成します。CRTDUPOBJ(QWBIISRSC) FROMLIB(QWBISVR44)OBJTYPE(*JOBD)TOLIB (QWBISVR44) NEWOBJ(newemailname)。ここで、newemailname は新規コネクターのジョブ記述に使用する 10 文字の名前です。

6. (i5/OS のみ) 新規コネクターを WebSphere Business Integration Server Express Console に追加します。WebSphere Business Integration Server Express Console の詳細については、このコンソールに付属のオンライン・ヘルプを参照してください。

第 6 章 コネクターの保守

本章では、アダプターによるエラー処理について説明します。この章の内容は、次のとおりです。

- 『コネクターのエラー処理』
- 42 ページの『トレース・メッセージ』

コネクターのエラー処理

コネクターは、トレース・レベルに関係なく、処理中に発生した異常状態をすべてログに記録します。コネクターは、エラー・テキストをコネクター・ログ・ファイルに書き込みます。このファイルの名前およびロケーションは、LogFileName コネクター構成プロパティーを使用して設定します。

メッセージには、状態および結果の詳細記述が含まれます。また、ビジネス・オブジェクト・ダンプやスタック・トレースなどのデバッグ時に補助となる追加情報が含まれる場合もあります (特例)。

エラー・メッセージの完全なリストについては、*ProductDir*¥connectors¥messages ディレクトリーにインストールされている BIA_TCPIPAdapter.txt メッセージ・ファイルを参照してください。表 6 に、よく見られるエラーのいくつかを示し、コネクターがこれらのエラーを処理する方法を説明します。

表 6. コネクターのエラー

エラーの説明	エラー・タイプ	エラー処理	修正手順
必須の CFG プロパティーにデータが読み込まれていない	致命的	エラーが記録され、コネクターは終了します。	プロパティーにデータが読み込まれていることを確認してください。
前提条件の BO 定義がリポジトリーに存在しない	致命的	エラーが記録され、コネクターは終了します。	すべての .xsd ファイルがリポジトリーに存在することを確認してください。
クライアントのみ、またはサーバーのみが CFG で構成される	警告	警告が記録され、コネクターはそれぞれ要求またはイベントのみを処理します。	両方の機能が必要な場合は、CFG のすべてのプロパティーが構成されていることを確認してください。
CFG プロパティーのタイプが不適切。例えば、正の値が必要な箇所に負の値が入っている	致命的	エラーが記録され、コネクターは終了します。	プロパティーのタイプが適切であることを確認してください。

表 6. コネクターのエラー (続き)

エラーの説明	エラー・タイプ	エラー処理	修正手順
要求に対して構成されているサーバーが使用できない	エラー	要求は失敗しますが、コネクターは終了しません。	次のことを確認します。a) アプリケーション・サーバーが稼働していること。b) リモート・ホストがコネクターのマシンから使用可能であること。
要求に対して構成されているサーバーのソケットが、要求が送信される前にタイムアウトになった	エラー	ソケット閉止エラーが記録されました。コネクターは継続されます。障害状況がブローカーに送信されました。	前述の項を参照。
完了イベントを受信する前にソケットが閉じられた	エラー	エラーが記録され、コネクターは動作を継続します。	コネクターにデータを送信した直後は、ソケットを閉じないでください。
ソケットとローカル・ポートのバインドを試行するときのバインド例外	致命的	エラーが記録され、コネクターは終了します。	ポートが空いていることを確認してください。
マップ MO に指定したマップが存在しない	エラー	エラーが記録され、コネクターは終了します。	マップの .xsd ファイルがリポジトリーで使用可能なことを確認してください。
PIMO レベル・エラー	エラー	PIMO インフラストラクチャーがエラーを記録し、コネクターは終了します。	マップと関連のアクションが適切に構成されていることを確認してください。

トレース・メッセージ

トレースは、コネクターの動作を細かく追跡するためにオンにすることができるオプションのデバッグ・フィーチャーです。トレース・メッセージは構成可能であり、動的変更が可能です。必要な詳細に応じて、さまざまなレベルを設定できます。次のセクションでは、TCP/IP アダプターのトレースについて説明します。

推奨: トレースは、パフォーマンスを向上させ、ファイル・サイズを小さくするために、実動システム上ではオフにするか、できるだけ低レベルに設定しておいてください。

コネクターでのトレースの使用

トレース・メッセージは、デフォルトでは「STDOUT」(画面) に書き込まれます。また、トレースをファイルに書き込むように構成することもできます。

表 7 に、各トレース・レベルでコネクターが出力する各種のトレース・メッセージを示します。すべてのトレース・メッセージがコネクター・プロパティー TraceFileName によって指定されたファイルに表示されます。これらのメッセージは、IBM WebSphere Business Integration Server Express アダプター・アーキテクチャーによって出力されるトレース・メッセージに追加されます。

表 7. コネクターのトレース・メッセージ

トレース・レベル	トレース・メッセージ
レベル 0	アダプターのバージョンをトレースします。
レベル 1	• pollForEvents メソッドが呼び出されるたびにトレースします。 • アダプターが TCP/IP サーバー・モードで新規のソケット接続を受け入れるたびにトレースします。 • アダプターが TCP/IP クライアント・モードで新規のソケット接続を試行するたびにトレースします。 • アダプターによって作成された ASBO/ISBO 名をトレースして、ブローカーに配信します。 • ブローカーによって作成された ASBO/ISBO 名の要求をトレースして、アダプターに配信します。
レベル 2	• doVerbFor () が呼び出されるたびにトレースを行います。この要求を処理しているプロトコル・ハンドラーをトレースします。 • executeCollab() または gotApplEvent() が呼び出されるたびにトレースします。
レベル 3	• 処理されているビジネス・オブジェクトの重要な ASI をトレースします。 • 処理されているビジネス・オブジェクトの重要な属性をトレースします。 • TCP チャネルを双方向に移動するすべてのデータをトレースします。 注: データの書式は 16 進数として設定されます。
レベル 4	• スレッドの生成をトレースします。 • 処理されたすべての ASI をトレースします。 • 重要な関数の入り口と出口をトレースします。
レベル 5	• 重要なメソッドごとに入り口と出口をトレースします。 • すべてのアダプター・プロパティーをトレースします。 • ブローカーに送信された BO のダンプをトレースします。 • ブローカーが送信した BO のダンプをトレースします。

付録 A. 標準構成プロパティー

この付録では、WebSphere Business Integration Server Express アダプターのコネクター・コンポーネントの標準構成プロパティーについて説明します。説明は、InterChange Server Express が対象となります。

このコネクターに固有のプロパティーについては、本書の該当するセクションを参照してください。

新規プロパティー

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。

- AdapterHelpName
- ControllerEventSequencing
- jms.ListenerConcurrency
- jms.TransportOptimized
- TivoliTransactionMonitorPerformance

標準コネクター・プロパティーの概要

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。フレームワークが使用します。
- アプリケーション固有またはコネクター固有の構成プロパティー。エージェントが使用します。

これらのプロパティーは、アダプターのフレームワークおよびエージェントの実行時の振る舞いを決定します。

このセクションでは、Connector Configurator Express の始動方法について説明し、すべてのプロパティーに共通する特性について説明します。コネクター固有の構成プロパティーについては、該当するアダプターのユーザーズ・ガイドを参照してください。

Connector Configurator Express の始動

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

Connector Configurator Express と System Manager は、Windows システム上でのみ動作します。コネクターを Linux 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。

Linux 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、Linux の統合ブローカーに接続してから、コネクター用の Connector Configurator Express を開く必要があります。

構成プロパティー値の概要

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。

1. デフォルト
2. InterChange Server Express 統合ブローカー用のリポジトリー
3. ローカル構成ファイル
4. コマンド行

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。STRING プロパティー・タイプの長さに制限はありません。INTEGER タイプの長さは、アダプターを実行しているサーバーによって決まります。

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。

プロパティーの更新特性 (すなわちコネクター・プロパティーへの変更を有効にする方法とタイミング) は、プロパティーの性質によって異なります。

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、新しい値が即時に有効になります。ただし、コネクターがスタンドアロン・モードの場合 (System Manager に依存しない) です。
- **エージェント再始動 (InterChange Server Express のみ)**
  コネクター・エージェントを停止して再始動しなければ、新規の値が有効になりません。
- **コンポーネント再始動**
  System Manager でコネクターを停止してから再始動しなければ、新しい値が有効になりません。エージェントまたはサーバー・プロセスを停止して再始動する必要はありません。
- **システム再始動**
  コネクター・エージェントおよびサーバーを停止して再始動しなければ、新規の値が有効になりません。

特定のプロパティーの更新方法を確認するには、「Connector Configurator Express」ウィンドウ内の「**更新メソッド**」列を参照するか、47 ページの表 8 の「更新メソッド」列を参照してください。

標準プロパティーが存在できる場所が 3 箇所あります。一部のプロパティーは複数の場所にあってもかまいません。

- **ReposController**
  このプロパティーはコネクター・コントローラー内にあり、その場所でのみ有効です。エージェント・サイドで値を変更した場合、コントローラーには影響しません。
- **ReposAgent**
  このプロパティーはエージェント内にあり、その場所でのみ有効です。プロパティーによっては、ローカル構成によってこの値をオーバーライドされることがあります。
- **LocalConfig**
  このプロパティーは、コネクターの構成ファイル内にあり、構成ファイルを通じてのみ機能することができます。コントローラーはこのプロパティーの値を変更することができず、システムが再配置されてコントローラーが明示的に更新されなければ、構成ファイルに加えられた変更を認識しません。

標準プロパティーの早見表

表 8 は、標準コネクター構成プロパティーの早見表です。すべてのコネクターでこれらのプロパティーすべてを必要とするわけではなく、プロパティー設定は異なる場合があります。

各プロパティーの説明については、表の次のセクションを参照してください。

注: 表 8 の注の欄で、「RepositoryDirectory が <REMOTE> に設定され」という句は、ブローカーが InterChange Server Express であることを示します。

表 8. 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
AdapterHelpName	有効な <Regional Setting> ディレクトリーを含む <ProductDir>¥bin¥Data¥App¥Help 内の有効なサブディレクトリーのいずれか	テンプレート名 (有効な場合) またはブランク・フィールド	コンポーネント再始動	サポートされる地域設定。chs_chn、cht_twn、deu_deu、esn_esp、fra_fra、ita_ita、jpn_jpn、kor_kor、ptb_bra、および enu_usa (デフォルト) を含む。
AdminInQueue	有効な JMS キュー名	*<CONNECTORNAME>*/ADMININQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
AdminOutQueue	有効な JMS キュー名	*<CONNECTORNAME>*/ADMINOUTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
AgentConnections	1 から 4	1	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が IDL であり、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が InterChange Server Express である場合のみ有効です。

表 8. 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
AgentTraceLevel	0 から 5	0	Inter-Change Server Express の場合は動的、その他の場合はコンポーネント再始動	
ApplicationName	アプリケーション名	コネクターのアプリケーション名に指定された値	コンポーネント再始動	
BrokerType	InterChange Server Express	InterChange Server Express	コンポーネント再始動	
CharacterEncoding	サポートされる任意のコード。次のリストはその一部です。ascii7、ascii8、SJIS、Cp949、GBK、Big5、Cp297、Cp273、Cp280、Cp284、Cp037、Cp437	ascii7	コンポーネント再始動	このプロパティーは、C++ コネクターでのみ有効です。
CommonEventInfrastructure	true または false	false	コンポーネント再始動	
CommonEventInfrastructureURL	URL ストリング。例えば、corbaloc:iiop:host:2809。	デフォルト値はありません。	コンポーネント再始動	このプロパティーは、CommonEvent Infrastructure の値が true の場合のみ有効です。
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント再始動	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が InterChange Server Express である場合のみ有効です。
ContainerManagedEvents	ブランクまたは JMS	ブランク	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
ControllerEventSequencing	true または false	true	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が InterChange Server Express である場合のみ有効です。
ControllerStoreAndForwardMode	true または false	true	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が InterChange Server Express である場合のみ有効です。
ControllerTraceLevel	0 から 5	0	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が InterChange Server Express である場合のみ有効です。
DeliveryQueue	任意の有効な JMS キュー名	*<CONNECTORNAME>*/DELIVERYQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
DeliveryTransport	IDL または JMS	RepositoryDirectory の値が <REMOTE> の場合は IDL。それ以外の場合は JMS。	コンポーネント再始動	RepositoryDirectory の値が <REMOTE> ではない場合、このプロパティーの有効な値は JMS のみです。

表 8. 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	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
EnableOidForFlowMonitoring	true または false	false	コンポーネント再始動	このプロパティーは、BrokerType の値が InterChange Server Express の場合のみ有効です。
FaultQueue	任意の有効なキュー名	*<CONNECTORNAME>*/FAULTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory、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	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
jms.ListenerConcurrency	1 から 32767	1	コンポーネント再始動	このプロパティーは、jms.TransportOptimized の値が true の場合のみ有効です。
jms.MessageBrokerName	FactoryClassName の値が IBM の場合は crossworlds.queue.manager を使用。	crossworlds.queue.manager	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
jms.Password	任意の有効なパスワード		コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
jms.TransportOptimized	true または false	false	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS で、BrokerType の値が InterChange Server Express である場合のみ有効です。
jms.UserName	任意の有効な名前		コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が InterChange Server Express である場合のみ有効です。
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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が InterChange Server Express である場合のみ有効です。
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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が InterChange Server Express である場合のみ有効です。

表 8. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
Locale	これは、サポートされるロケールの一部です。 en_US、ja_JP、ko_KR、zh_CN 、zh_TW、fr_FR、de_DE、it_IT /xph>、es_ES、pt_BR	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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が InterChange Server Express である場合のみ有効です。
MaxEventCapacity	1 から 2147483647	2147483647	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が InterChange Server Express である場合のみ有効です。
MessageFileName	有効なファイル名	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	このプロパティーは、DuplicateEventElimination の値が true で、ContainerManagedEvents に値がない場合にのみ有効です。
OADAutoRestartAgent	true または false	false	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が InterChange Server Express である場合のみ有効です。
OADMaxNumRetry	正整数	1000	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が InterChange Server Express である場合のみ有効です。
OADRetryTimeInterval	正整数 (単位: 分)	10	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が InterChange Server Express である場合のみ有効です。
PollEndTime	HH = 0 から 23 MM = 0 から 59	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒)	10000	ブローカーが InterChange Server Express の場合は動的、その他の場合はコンポーネント再始動	
PollQuantity	1 から 500	1	エージェント再始動	このプロパティーは、ContainerManagedEvents の値が JMS の場合のみ有効です。
PollStartTime	HH = 0 から 23 MM = 0 から 59	HH:MM	コンポーネント再始動	

表 8. 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
RepositoryDirectory	ブローカーが InterChange Server Express である場合は <REMOTE>、その他の場合は任意の有効なローカル・ディレクトリー。	InterChange Server Express の場合、値は <REMOTE> に設定されます。	エージェント再始動	
RequestQueue	有効な JMS キュー名	*<CONNECTORNAME>*/REQUESTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
ResponseQueue	有効な JMS キュー名	*<CONNECTORNAME>*/RESPONSEQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
RestartRetryCount	0 から 99	3	InterChange Server Express の場合は動的、その他の場合はコンポーネント再始動	
RestartRetryInterval	1 から 2147483647 までの値 (分単位)。	1	InterChange Server Express の場合は動的、その他の場合はコンポーネント再始動	
RHF2MessageDomain	mrmまたは xml	mrm	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS で、WireFormat の値が CwXML の場合のみ有効です。
SourceQueue	任意の有効な WebSphere MQ キュー名	*<CONNECTORNAME>*/SOURCEQUEUE	エージェント再始動	このプロパティーは、ContainerManagedEvents の値が JMS の場合のみ有効です。
SynchronousRequest Queue	任意の有効なキュー名	*<CONNECTORNAME>*/SYNCHRONOUSREQUEST QUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
SynchronousRequest Timeout	0 から任意の数 (ミリ秒)	0	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
SynchronousResponse Queue	任意の有効なキュー名	*<CONNECTORNAME>*/SYNCHRONOUSRESPONSE QUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
TivoliMonitorTransaction Performance	true または false	false	コンポーネント再始動	
WireFormat	CwXMLまたは CwBO	CwXML	エージェント再始動	RepositoryDirectory の値が <REMOTE> に設定されていない場合、このプロパティーの値は、CwXML でなければなりません。RepositoryDirectory の値が <REMOTE> に設定されている場合、値は CwBO でなければなりません。
WsifSynchronousRequest Timeout	0 から任意の数 (ミリ秒)	0	コンポーネント再始動	BrokerType の値が InterChange Server Express の場合、このプロパティーは無効です。

表 8. 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
XMLNameSpaceFormat	shortまたは long	short	エージェント再始動	BrokerType の値が InterChange Server Express の場合、このプロパティーは無効です。

標準プロパティー

このセクションでは、標準コネクター構成プロパティーについて説明します。

AdapterHelpName

AdapterHelpName プロパティーは、コネクター固有の全般ヘルプ・ファイルがあるディレクトリーの名前です。ディレクトリーは、*<ProductDir>*¥bin¥Data¥App¥Help 内に配置される必要があり、少なくとも言語ディレクトリー enu_usa が含まれていなければなりません。ロケールに応じて、その他のディレクトリーが含まれることがあります。

デフォルト値は、テンプレート名が有効であればテンプレート名、有効でなければブランクです。

AdminInQueue

AdminInQueue プロパティーは、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を指定します。

デフォルト値は *<CONNECTORNAME>*/ADMININQUEUE です。

AdminOutQueue

AdminOutQueue プロパティーは、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を指定します。

デフォルト値は *<CONNECTORNAME>*/ADMINOUTQUEUE です。

AgentConnections

AgentConnections プロパティーは、ORB (オブジェクト・リクエスト・ブローカー) が初期化するときに開かれる ORB 接続の数を制御します。

このプロパティーのデフォルト値は 1 です。

AgentTraceLevel

AgentTraceLevel プロパティーは、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を設定します。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。

デフォルト値は 0 です。

ApplicationName

ApplicationName プロパティーは、コネクター・アプリケーションの名前を一意的に識別します。この名前は、システム管理者が統合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。

デフォルトはコネクターの名前です。

BrokerType

BrokerType プロパティーは、使用している統合ブローカーのタイプを識別します。値は `ICS (InterChange Server Express)` です。

CharacterEncoding

CharacterEncoding プロパティーは、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、このプロパティーに `ascii7` という値が使用されます。

デフォルトでは、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。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 (*<ProductDir>*) にある ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。詳細については、本書の付録『Connector Configurator Express』を参照してください。

ConcurrentEventTriggeredFlows

ConcurrentEventTriggeredFlows プロパティーは、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されるビジネス・オブジェクトの数に設定します。例えば、このプロパティーの値を `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。

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、以下のプロパティーを構成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、複数のスレッドを使用できるようにコラボレーションを構成する必要があります。
- 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を、複数の要求を並行して処理できるように構成する必要があります。

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。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 1 です。

ContainerManagedEvents

ContainerManagedEvents 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、1 つの JMS トランザクションとして宛先キューに配置されます。

このプロパティーを 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設定する必要があります。

- PollQuantity = 1 から 500
- SourceQueue = /SOURCEQUEUE

また、MimeType および DHClass (データ・ハンドラー・クラス) 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。DataHandlerConfigMOName (オプションのメタオブジェクト名) を追加することもできます。これらのプロパティーの値を設定するには、Connector Configurator Express の「**データ・ハンドラー**」タブを使用します。

これらのプロパティーはアダプター固有ですが、以下に値の例をいくつか示します。

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

「**データ・ハンドラー**」 タブのこれらの値のフィールドは、ContainerManagedEvents プロパティーを JMS という値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。

ContainerManagedEvents プロパティーは、DeliveryTransport プロパティーの値が JMS に設定されている場合のみ有効です。

デフォルト値はありません。

ControllerEventSequencing

ControllerEventSequencing プロパティーは、コネクター・コントローラーでイベント順序付けを使用可能にします。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType が InterChange Server Express) のみ有効です。T

デフォルト値は true です。

ControllerStoreAndForwardMode

ControllerStoreAndForwardMode プロパティーは、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 InterChange Server Express (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。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType プロパティーの値が InterChange Server Express) のみ有効です。

デフォルト値は true です。

ControllerTraceLevel

ControllerTraceLevel プロパティーは、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を設定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 0 です。

DeliveryQueue

DeliveryQueue プロパティーは、コネクターが統合ブローカーへビジネス・オブジェクトを送信するときに使用するキューを定義します。

このプロパティーは、DeliveryTransport プロパティーの値が JMS に設定されている場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/DELIVERYQUEUE です。

DeliveryTransport

DeliveryTransport プロパティーは、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。Java Messaging Service の場合、値は JMS です。

- RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合、DeliveryTransport プロパティーの値には IDL または JMS を使用することができ、デフォルトは IDL です。

RepositoryDirectory プロパティーの値が IDL である場合、コネクターは、CORBA IIOP を使用してサービス呼び出し要求と管理メッセージを送信します。

JMS

JMS トランスポート機構は、Java Messaging Service (JMS) を使用した、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 Express 内にリストされます。jms.MessageBrokerName プロパティーおよび jms.FactoryClassName プロパティーは、このトランスポートの必須プロパティーです。

InterChange Server Express (ICS) が統合ブローカーである場合、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。

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・サイドの) コネクター・コントローラーと (クライアント・サイドの) 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 768MB 未満である場合には、次の変数およびプロパティーを設定してください。

- 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 (<*ProductDir*>) 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、Windows 版、Linux 版、または i5/OS 版の「*WebSphere Business Integration Server Express インストール・ガイド*」を参照してください。

DuplicateEventElimination

このプロパティーの値が true の場合、JMS 対応コネクターでは重複イベントがデリバリー・キューへデリバリーされないようにすることができます。この機能を使

用するには、コネクター開発時に、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。

注: このプロパティーの値が `true` の場合、保証付きイベント・デリバリーを提供するには、MonitorQueue プロパティーを使用可能にする必要があります。

デフォルト値は `false` です。

EnableOidForFlowMonitoring

このプロパティーの値が `true` の場合、アダプター・ランタイムは、着信 ObjectEventID にフロー・モニターの外部キーのマークを付けます。

このプロパティーは、BrokerType プロパティーが InterChange Server Express に設定されている場合のみ有効です。

デフォルト値は `false` です。

FaultQueue

コネクターでメッセージを処理中にエラーが発生すると、コネクターは、そのメッセージ (および状況標識と問題説明) を FaultQueue プロパティーで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。

jms.FactoryClassName

jms.FactoryClassName プロパティーは、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。DeliveryTransport プロパティーの値が `JMS` に設定されている場合、このプロパティーを設定する必要があります。

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

jms.ListenerConcurrency

jms.ListenerConcurrency プロパティーは、JMS コントローラーの並行リスナーの数を指定します。コントローラー内部で、並行してメッセージを取り出して処理するスレッドの数を指定します。

このプロパティーは、jms.OptimizedTransport プロパティーの値が `true` の場合のみ有効です。

デフォルト値は `1` です。

jms.MessageBrokerName

jms.MessageBrokerName は、JMS プロバイダーのために使用するブローカー名を指定します。`JMS` をデリバリー・トランスポート機構として (DeliveryTransport プロパティーで) 指定する場合、このコネクター・プロパティーを設定する必要があります。

リモート・メッセージ・ブローカーに接続した場合、このプロパティーでは以下の値を指定する必要があります。
QueueMgrName:*Channel*:*HostName*:*PortNumber*
ここで、以下のように説明されます。
QueueMgrName は、キュー・マネージャー名です。
Channel は、クライアントが使用するチャネルです。
HostName は、キュー・マネージャーの配置先のマシン名です。
PortNumber は、キュー・マネージャーが listen に使用するポートの番号です。

例えば、次のようにします。
```
jms.MessageBrokerName = WBIMB.Queue.Manager:CHANNEL1:RemoteMachine:1456
```

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

jms.NumConcurrentRequests

jms.NumConcurrentRequests プロパティーは、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はブロックされ、処理を続行するには他のいずれかの要求が完了するのを待機する必要があります。

デフォルト値は `10` です。

jms.Password

jms.Password プロパティーは、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。

デフォルト値はありません。

jms.TransportOptimized

jms.TransportOptimized プロパティーは、WIP (処理中の作業) が最適化されるかどうかを決定します。WIP を最適化するには、WebSphere MQ プロバイダーが必要です。最適化された WIP が作動するためには、メッセージング・プロバイダーが以下の操作を実行できなければなりません。

1. メッセージをキューから削除せずに読み取る。
2. メッセージ全体を受信側のメモリー空間に転送することなく、固有の ID を使用してメッセージを削除する。
3. 固有の ID を使用してメッセージを読み取る (リカバリーのために必要)。
4. 読み取られなかったイベントが現れるポイントを追跡する。

JMS API は、上記の条件 2 および 4 を満たさないため、最適化された WIP には使用できませんが、MQ Java API は 4 つの条件をすべて満たすため、最適化された WIP には必要です。

このプロパティーは、DeliveryTransport の値が `JMS` で、BrokerType の値が `InterChange Server Express` である場合のみ有効です。

デフォルト値は `false` です。

jms.UserName

jms.UserName プロパティーは、JMS プロバイダー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。

デフォルト値はありません。

JvmMaxHeapSize

JvmMaxHeapSize プロパティーは、エージェントの最大ヒープ・サイズ (メガバイト単位) を指定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 128M です。

JvmMaxNativeStackSize

JvmMaxNativeStackSize プロパティーは、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) を指定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 128K です。

JvmMinHeapSize

JvmMinHeapSize プロパティーは、エージェントの最小ヒープ・サイズ (メガバイト単位) を指定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 1M です。

Locale

Locale プロパティーは、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 は、2 文字の言語コード (小文字を使用) です。
TT は、2 文字の国または地域コード (大文字を使用) です。
codeset は、関連文字コード・セットの名前です (オプションの場合があります)。

デフォルトでは、サポートされるロケールの一部のみがリストされます。サポートされる他の値をリストに追加するには、<*ProductDir*>¥bin ディレクトリーにある

¥Data¥Std¥stdConnProps.xml ファイルを変更します。詳細については、本書の付録『Connector Configurator Express』を参照してください。

コネクターが国際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、そのアダプターのユーザーズ・ガイドを参照してください。

デフォルト値は en_US です。

LogAtInterchangeEnd

LogAtInterchangeEnd プロパティーは、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。

ログ宛先にログを記録すると、E 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 ファイルで MESSAGE_RECIPIENT の値として指定された宛先に対する E メール・メッセージが生成されます。例えば、LogAtInterChangeEnd の値を 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、E メール・メッセージが送信されます。

このプロパティーは、RespositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が InterChange Server Express) のみ有効です。

デフォルト値は false です。

MaxEventCapacity

MaxEventCapacity プロパティーは、コントローラー・バッファー内のイベントの最大数を指定します。このプロパティーは、フロー制御機能によって使用されます。

このプロパティーは、RespositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が InterChange Server Express) のみ有効です。

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

MessageFileName プロパティーは、コネクター・メッセージ・ファイルの名前を指定します。メッセージ・ファイルの標準位置は、製品ディレクトリーの ¥connectors¥messages です。メッセージ・ファイルが標準位置に格納されていない場合は、メッセージ・ファイル名を絶対パスで指定します。

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

注: 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。

デフォルト値は InterchangeSystem.txt です。

MonitorQueue

MonitorQueue プロパティーは、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を指定します。

このプロパティーは、DeliveryTransport プロパティーの値が JMS で、DuplicateEventElimination の値が true の場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/MONITORQUEUE です。

OADAutoRestartAgent

OADAutoRestartAgent プロパティーは、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、WebSphere 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。WebSphere MQ-triggered OAD 機能の構成方法については、「*WebSphere Business Integration Server Express インストール・ガイド Windows 版*」、「*WebSphere Business Integration Server Express インストール・ガイド Linux 版*」、または「*WebSphere Business Integration Server Express インストール・ガイド i5/OS 版*」を参照してください。

このプロパティーは、RespositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が InterChange Server Express) のみ有効です。

デフォルト値は false です。

OADMaxNumRetry

OADMaxNumRetry プロパティーは、異常シャットダウンの後で WebSphere MQ によりトリガーされる Object Activation Daemon (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。

このプロパティーは、RespositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が InterChange Server Express) のみ有効です。

デフォルト値は 1000 です。

OADRetryTimeInterval

OADRetryTimeInterval プロパティーは、WebSphere MQ によりトリガーされる Object Activation Daemon (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。

このプロパティーは、RespositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が InterChange Server Express) のみ有効です。

デフォルト値は `10` です。

PollEndTime

PollEndTime プロパティーは、イベント・キューのポーリングを停止する時刻を指定します。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、値を含まない `HH:MM` であるため、この値は必ず変更する必要があります。

アダプター・ランタイムが以下のことを検出した場合、

- PollStartTime が設定されて、PollEndTime が設定されていない、または
- PollEndTime が設定されて、PollStartTime が設定されていない

PollFrequency プロパティーに構成された値を使用してポーリングします。

PollFrequency

PollFrequency プロパティーは、あるポーリング・アクションの終了から次のポーリング・アクションの開始までの時間をミリ秒単位で指定します。これはポーリング・アクションの間隔ではありません。正確には、次のような論理になります。

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

このプロパティーでは、以下の値が有効です。

- ポーリング・アクション間のミリ秒数 (正整数)。
- 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。
- 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。

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。制限が存在する場合、これらの制約事項はアダプターのインストールと構成の章に説明されています。

PollQuantity

PollQuantity プロパティーは、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。

このプロパティーは、DeliveryTransport プロパティーの値が JMS で、ContainerManagedEvents プロパティーに値がある場合のみ有効です。

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

- 一度ポーリングされると、コネクターはメッセージの本文を検出し、それを添付ファイルとして読み取ります。本文の MIME タイプにはデータ・ハンドラーが指定されていないので、コネクターはメッセージを無視します。
- コネクターは最初の BO 添付ファイルを処理します。この MIME タイプには対応するデータ・ハンドラーがあるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
- 2 回目のポーリングを受けると、コネクターは BO の 2 番目の添付を処理します。この MIME タイプには対応するデータ・ハンドラーがあるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
- それが受け入れられると、3 番目の BO 添付ファイルが送信されます。

PollStartTime

PollStartTime プロパティーは、イベント・キューのポーリングを開始する時刻を指定します。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、値を含まない HH:MM であるため、この値は必ず変更する必要があります。

アダプター・ランタイムが以下のことを検出した場合、

- PollStartTime が設定されて、PollEndTime が設定されていない、または
- PollEndTime が設定されて、PollStartTime が設定されていない

PollFrequency プロパティーに構成された値を使用してポーリングします。

RepositoryDirectory

RepositoryDirectory プロパティーは、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保管されています。

統合ブローカーが InterChange Server Express 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 Express 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、この値はデフォルトで <*ProductDir*>¥repository に設定されます。ただし、この値には任意の有効なディレクトリー名を設定できます。

RequestQueue

RequestQueue プロパティーは、統合ブローカーが、ビジネス・オブジェクトをコネクターに送信するときに使用されるキューを指定します。

このプロパティーは、DeliveryTransport プロパティーの値が JMS の場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/REQUESTQUEUE です。

ResponseQueue

ResponseQueue プロパティーは、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 InterChange Server Express (InterChange Server Express) 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。

このプロパティーは、DeliveryTransport プロパティーの値が JMS の場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/RESPONSEQUEUE です。

RestartRetryCount

RestartRetryCount プロパティーは、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に接続された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クライアント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

RestartRetryInterval プロパティーは、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にリンクされた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クライアント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。

プロパティーに使用可能な値の範囲は 1 から 2147483647 です。

デフォルト値は 1 です。

RHF2MessageDomain

RHF2MessageDomain プロパティーにより、JMS ヘッダーのドメイン名フィールドの値を構成できます。JMS トランスポートを介してデータを WebSphere Message Broker に送信するときに、アダプター・フレームワークにより JMS ヘッダー情報、ドメイン名、および固定値 mrm が書き込まれます。構成可能ドメイン名によって、WebSphere Message Broker がメッセージ・データを処理する方法を追跡できます。

ヘッダーの例を示します。

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

BrokerType の値が InterChange Server Express の場合、このプロパティーは無効です。また、このプロパティーは、DeliveryTransport プロパティーの値が JMS で、WireFormat プロパティーの値が CwXML の場合のみ有効です。

可能な値は、mrm および xml です。デフォルト値は mrm です。

SourceQueue

SourceQueue プロパティーは、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、54 ページの『ContainerManagedEvents』を参照してください。

このプロパティーは、DeliveryTransport の値が JMS で、ContainerManagedEvents の値が指定されている場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/SOURCEQUEUE です。

SynchronousRequestQueue

SynchronousRequestQueue プロパティーは、同期応答を要求する要求メッセージを、コネクター・フレームワークからブローカーに配信します。このキューは、コネクターが同期実行を使用する場合にのみ必要です。同期実行の場合、コネクター・フレームワークは、同期要求キューにメッセージを送信し、同期応答キューでブローカーからの応答を待機します。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。

このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/SYNCHRONOUSREQUESTQUEUE です。

SynchronousRequestTimeout

SynchronousRequestTimeout プロパティーは、コネクターが同期要求への応答を待機する時間をミリ秒単位で指定します。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 (およびエラー・メッセージ) を障害キューに移動します。

このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。

デフォルト値は 0 です。

SynchronousResponseQueue

SynchronousResponseQueue プロパティーは、同期要求に対す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。

このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。

デフォルトは *<CONNECTORNAME>*/SYNCHRONOUSRESPONSEQUEUE です。

TivoliMonitorTransactionPerformance

TivoliMonitorTransactionPerformance プロパティーは、IBM Tivoli Monitoring for Transaction Performance (ITMTP) を実行時に起動するかどうかを指定します。

デフォルト値は false です。

WireFormat

WireFormat プロパティーは、トランスポートのメッセージ・フォーマットを指定します。

- RepositoryDirectory プロパティーの値がローカル・ディレクトリーの場合、値は CwXML です。
- RepositoryDirectory プロパティーの値がリモート・ディレクトリーの場合、値は CwBO です。

付録 B. コネクター固有のプロパティーおよび必須のビジネス・オブジェクト・プロパティー

表 9 には、Adapter for TCP/IP に固有のプロパティーおよび値の要約を示します。正符号 (+) は、子プロパティーを示します。各プロパティーの定義および説明は、表の後に記載します。これらの値の設定には、Connector Configurator を使用します。

表 9. コネクター固有のプロパティー

プロパティー	指定可能な値	デフォルト値	更新メソッド	必須ですか
ServerConfiguration			エージェント再始動	いいえ。ただし、この設定を取り込まない場合、イベント処理は使用可能になりません。警告メッセージがログに記録されます。
+Port	1 以上の整数		エージェント再始動	はい
+TransportProtocol	このリリースでは、tcp のみがサポートされます。将来は、「secure tcp/ip」など、その他の値も可能になります。	tcp	エージェント再始動	はい
+MaxRequest Processors	1 以上の整数	2	エージェント再始動	いいえ
+MaxRequest PoolSize	1 以上の整数	3	エージェント再始動	いいえ
+ServerQueueLength	1 以上の整数		エージェント再始動	いいえ
+ReceiveBufferSize	1 以上の整数		エージェント再始動	いいえ
+SendBufferSize	1 以上の整数		エージェント再始動	いいえ

表 9. コネクター固有のプロパティー (続き)

プロパティー	指定可能な値	デフォルト値	更新メソッド	必須ですか
+KeepAlive	true または false	false	エージェント再始動	いいえ
+ServerSocket Timeout	1 以上の整数		コネクターの再始動	いいえ
+SocketTimeout	1 以上の整数		エージェント再始動	いいえ
+RetryInterval	0 以上の整数	0	エージェント再始動	いいえ
+NumberofRetries	0 以上の整数	0	エージェント再始動	いいえ
ClientConfiguration			エージェント再始動	いいえ。ただし、この設定を取り込まない場合、サービス呼び出し要求処理は使用可能になりません。警告メッセージがログに記録されます。
+Clients			エージェント再始動	はい。これは、特定の子クライアント・データを保持する階層プロパティーです。この設定を取り込まない場合、警告のログが記録されます。
++Client1	クライアント名		エージェント再始動	いいえ
+++Host	リモート・ホストのアドレス		エージェント再始動	はい
+++Port	リモート・ホストのポート番号		エージェント再始動	はい
+++TransportProtocol	このハンドラーが実装しているトランスポート・プロトコル	tcp	エージェント再始動	はい

表 9. コネクター固有のプロパティー (続き)

プロパティー	指定可能な値	デフォルト値	更新メソッド	必須ですか
+++ReceiveBuffer Size	1 以上の整数		エージェント再始動	いいえ
+++SendBufferSize	1 以上の整数		エージェント再始動	いいえ
+++KeepAlive	true または false	false	エージェント再始動	いいえ
+++SocketTimeout	1 以上の整数		エージェント再始動	いいえ
+++MaxAttempts ToRead	1 以上の整数	1	エージェント再始動	いいえ
+++RetryInterval	0 以上の整数	0	コネクターの再始動	いいえ
+++Numberof Retries	0 以上の整数	0	エージェント再始動	いいえ
++Client2	同上		エージェント再始動	いいえ
Configuration MetaObject	静的メタオブジェクト	BIA _Static _MO	エージェント再始動	はい
Service RegistrationMO	サービス・メタオブジェクト	BIA _MO _Service	エージェント再始動	はい
DataHandler MimeType	MIME タイプ	text/ name value	コネクターの再始動	はい
DataHandler MetaObjectName	DataHandler 構成メタオブジェクト	BIA _MO _Data Handler _Default	エージェント再始動	はい

コネクター固有の構成プロパティー

次に示すのは、前述のプロパティーの一連の定義です。

ServerConfiguration

イベント処理またはインバウンド処理のために TCP/IP コネクターが使用する 1 組のプロパティーです。この場合、コネクターは TCP サーバーとして動作し、定義されているポートの要求を listen します。1 つのコネクターに定義できるサーバーは 1 つのみです。

Port

コネクターによる listen の対象となるローカル・ポートです。

TransportProtocol

この listener が実装しているトランスポート・プロトコルです。このリリースでは、使用できる唯一の値は「tcp」です。将来のリリースでは、「secure TCP/IP」など、他の値が追加される可能性があります。

MaxRequestProcessors

定義されているポートの着信要求を同時に処理するスレッドの最大数を設定します。

MaxRequestPoolSize

同時に処理するためにキャッシュに格納する着信要求の最大数を設定します。任意のある瞬間に、コネクターは最大で (MaxRequestProcessors + MaxRequestPoolSize) 件の要求を処理できます。

ServerQueueLength

着信接続要求のサーバー・ソケット・キューの長さを設定します。この値は、ホストが接続の拒否を開始するまでに、一度に格納できる着信要求の数を指定します。

注: キューの最大長は、オペレーティング・システムに依存します。

ReceiveBufferSize

推奨されているネットワーク入出力バッファーのサイズを設定します。この値は、下位層にあるプラットフォームのネットワーキング・コードのヒントになります。バッファー・サイズを増加すると、大量の接続に対するネットワーク入出力のパフォーマンスが向上しますが、バッファー・サイズを縮小すると、着信データのバックログ削減に効果があります。

SendBufferSize

推奨されているネットワーク入出力バッファーのサイズを設定します。この値は、下位層にあるプラットフォームのネットワーキング・コードのヒントになります。バッファー・サイズを増加すると、大量の接続に対するネットワーク入出力のパフォーマンスが向上しますが、バッファー・サイズを縮小すると、着信データのバックログ削減に効果があります。

KeepAlive

ハートビートのプローブです。空のデータ・パケットに現行シーケンス、確認通知、およびウィンドウ番号を付加して、定期的に送信します。

ServerSocketTimeout

この ServerSocket のタイムアウト・ブロッキングをミリ秒単位で設定します。タイムアウトをゼロに設定すると、無限大のタイムアウトと解釈されます。このオプションをゼロ以外のタイムアウト値に設定すると、この ServerSocket の accept() の呼び出しは、この長さの時間だけ妨害されます。タイムアウト時間が超過すると、`java.io.InterruptedIOException` が発生します。ただし、ServerSocket は有効なままです。このオプションは、妨害操作を有効にする前に使用可能にする必要があります。

listener スレッドは、この間隔に要求を受信しなかった場合、Connector shutdown フラグが設定されているかどうかを検査します。Connector shutdown フラグが設定されている場合、listener スレッドは終了します。この値が適用されるのは、コネクターが要求を受け入れる TCP サーバーとして動作している場合のみです。

SocketTimeOut

ソケットの基本タイムアウト・ブロッキングをミリ秒単位で設定します。このオプションをゼロ以外のタイムアウト値に設定すると、このソケットの read() の呼び出しは、この長さの時間だけ妨害されます。タイムアウト時間が超過すると、`java.io.InterruptedIOException` が発生します。ただし、ソケットは有効なままです。このオプションは、妨害操作を有効にする前に使用可能にする必要があります。タイムアウトをゼロに設定すると、無限大のタイムアウトと解釈されます。

RetryInterval

TCP サーバー・モードのコネクターが、失敗した操作を再試行するまで待機する推奨の間隔を設定します。このような状態では、接続の受諾、読み取り/書き込みのためのストリームの開始、これらのストリームに対する読み取りまたは書き込みなどの間にエラーが発生している場合があります。

NumberofRetries

前述のエラー条件でサーバーが実行する再試行の推奨回数を設定します。

ClientConfiguration

サービス呼び出し要求処理またはアウトバウンド処理のために TCP/IP コネクターが使用する 1 組のプロパティーです。この場合、コネクターは TCP クライアントとして動作し、構成で定義されたリモート・ホストとの接続を開始します。1 つのコネクターに対して複数のクライアントを定義できます。

Clients

これは、クライアント構成を定義する子を保持するためにのみ機能する階層プロパティーです。

Client1

クライアントの名前を指定します。構成メタオブジェクトに指定されている ASI と互いに関係があります。

Host

リモート・ホストのアドレスを設定します。

Port

クライアントが接続先にする必要があるリモート・ホスト・ポートを設定します。

TransportProtocol

サポートされているトランスポート・プロトコルを設定します。このリリースでは、「tcp」がサポートされている唯一の値です。

ReceiveBufferSize

推奨されているネットワーク入出力バッファーのサイズを設定します。この値は、下位層にあるプラットフォームのネットワーキング・コードのヒントになります。バッファー・サイズを増加すると、大量の接続に対するネットワーク入出力のパフォーマンスが向上しますが、バッファー・サイズを縮小すると、着信データのバックログ削減に効果があります。

SendBufferSize

推奨されているネットワーク入出力バッファーのサイズを設定します。この値は、下位層にあるプラットフォームのネットワーキング・コードのヒントになります。バッファー・サイズを増加すると、大量の接続に対するネットワーク入出力のパフォーマンスが向上しますが、バッファー・サイズを縮小すると、着信データのバックログ削減に効果があります。

KeepAlive

ハートビートのプローブです。空のデータ・パケットに現行シーケンス、確認通知、およびウィンドウ番号を付加して、定期的に送信します。

SocketTimeout

このソケットのタイムアウト・ブロッキングをミリ秒単位で設定します。このプロパティーをゼロ以外の値に設定すると、このソケットに関連した InputStream に対する `read()` 呼び出しが、この長さの時間だけ妨害されます。タイムアウト時間が超過すると、`java.io.InterruptedIOException` が発生します。ただし、ソケットは有効なままです。このオプションは、妨害操作を有効にする前に使用可能にする必要があります。タイムアウトをゼロに設定すると、無限大のタイムアウトと解釈されます。

MaxAttemptsToRead

データの受信開始後、コネクターがソケットからデータを読み取る最大回数を設定します。このプロパティーを設定すると、別個の確認通知データの受信も可能にな

ります。この方法は、イベント処理とは異なります。イベント処理では、確認通知として受信するデータは少量であることが前提だからです。

RetryInterval

TCP クライアント・モードのコネクターが、失敗した操作を再試行するまで待機する推奨の間隔を設定します。このような状態では、読み取り/書き込みのためのストリームの開始、これらのストリームに対する読み取りまたは書き込みなどの間にエラーが発生している場合があります。

NumberofRetries

前述のエラー条件で TCP クライアント・モードのコネクターが実行する再試行の推奨回数を設定します。

Client2

次のクライアント構成の名前です。

ConfigurationMetaObject

静的構成情報を保持するメタオブジェクトです。 11 ページの『汎用メタオブジェクト』の `BIA_Static_MO` の説明を参照してください。

ServiceRegistrationMO

サービス情報を保持するトップレベルのメタオブジェクトです。 11 ページの『汎用メタオブジェクト』の `BIA_MO_Service` の説明を参照してください。

DataHandlerMimeType

着信データの予想 MIME タイプを設定します。適切な DataHandler を指定するために使用します。

DataHandlerMetaObjectName

サービス・オブジェクトに格納されている以外の DataHandler 構成情報を保持するトップレベルのメタオブジェクトです。 11 ページの『汎用メタオブジェクト』の `BIA_MO_DataHandler_Default` 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

「サポートされているビジネス・オブジェクト」タブの設定にも Connector Configurator を使用します。表 10 には、HL7 を処理する場合に出現する値を示します。その他の状態では、他の値が出現する場合があります。

表 10. サポートされているビジネス・オブジェクトのプロパティー

ビジネス・オブジェクト名	メッセージ・セット ID
BIA_ApplicationMessage	1
BIA_FinalMessage	2
BIA_MO_DataHandler_Default	3
BIA_MO_Service	4

表 10. サポートされているビジネス・オブジェクトのプロパティー (続き)

ビジネス・オブジェクト名	メッセージ・セット ID
BIA_MO_Tcpip_MapSubscriptions	5
BIA_ResponseMessage	6
BIA_STATIC_MO	7
BIA_InputMessage	8
HL7_SGMSH	9
HL7_MESSAGE	10

これらのオブジェクトの詳細については、9 ページの『内部ビジネス・オブジェクト』を参照してください。

付録 C. アダプターの処理

この付録では、PIMO インフラストラクチャーを含む TCP/IP アダプター内部の処理フローを詳細に説明します。

TCP/IP プロトコル

TCP/IP は、任意の種類のデータを異種の物理的ネットワークを介して送信するための一連のプロトコルです。このプロトコルは、インターネットを始め、その他の多くの大規模および小規模ネットワークの基礎になっています。データは、ステートレス IP プロトコルによりネットワークを介して配送されるパケットにカプセル化されます。IP パケットのストリームは、TCP (Transmission Control Protocol) によって制御されます。TCP の役目は、送信側ホストと受信側ホストとの間に「バーチャル・サーキット」を確立し、信頼性の高い伝送を実現することです。TCP プロトコルが実装されているソフトウェアは、2 種類あります。1 つは TCP クライアントで、接続を開始します。もう 1 つは TCP サーバーで、開始要求に対して割り当てられているポートを listen して、要求を受け入れます。TCP の構造は本質的に全二重です。つまり、TCP クライアントと TCP サーバーの両方がデータを同時に送受信できます。

Adapter for TCP/IP

Websphere Business Integration Server Express Adapter for TCP/IP は、未加工の TCP/IP 接続を介してシステムの内外に送信されるデータのルーティング方法を提供します。医療業界における HL7 プロトコルのように、ドメイン固有のメッセージを送信するための特定の業界標準プロトコルは、TCP/IP 接続を介して直接送信するように設計されています。Adapter for TCP/IP は、このようなデータを収集し、より大規模な統合フローに統合するための方法を提供します。

接続管理

TCP/IP Adapter のランタイム・コンポーネントであるコネクターは、TCP サーバー (インバウンド・モード、つまりイベント処理モード時) または TCP クライアント (アウトバウンド・モード、つまりサービス要求モード時) のいずれかとして機能します。実際の TCP プロトコルを実装しているコネクターのモジュールのことを接続マネージャーと呼びます。

接続マネージャーには、Protocol Listener フレームワークと Protocol Handler フレームワークの 2 つの部分があります。Protocol Listener フレームワークは、コネクター構成ファイル (CFG) の `ServerConfiguration` セクションのプロパティー設定に基づいて、TCP サーバーとして機能します。CFG の詳細については、67 ページの『付録 B. コネクター固有のプロパティーおよび必須のビジネス・オブジェクト・プロパティー』を参照してください。Protocol Listener フレームワークは、着信要求を listen して、Request Pool と呼ばれるスレッド管理コンポーネントを介して複数の要求の並列処理機能を提供します。すべての新規着信要求は、Request Pool で新規の Request オブジェクトになります。Request Pool のサイズは、構成プロパティ

ー MaxRequestPoolSize によって設定されます。Request Pool 内の要求は、CFG の MaxRequestProcessors プロパティーによって構成された数まで、作業者スレッドで処理されます。任意のある時点で、コネクターは最大で (MaxRequestPoolSize + MaxRequestProcessors) 件の要求を処理できます。ここでも、スレッド管理とロード・バランシングが考慮されます。作業者スレッドは、着信要求をバイト配列にパッケージ化し、それを Message Processing フレームワークに渡します。

Protocol Handler フレームワークは、TCP クライアントとして機能し、CFG の ClientConfiguration 部分に設定されているプロパティーに基づいて、サービス呼び出し要求データを受け取り、それをリモート・ホストに送信します。

メッセージ処理の管理

Message Processing フレームワークは、着信イベントのデータを使用可能なビジネス・オブジェクトへ変換する操作、および発信サービス要求ビジネス・オブジェクトを、TCP/IP を介して送信可能な、さまざまなサポートされるメッセージ構造へ変換する操作を管理します。Message Processing フレームワークは、PIMO フレームワーク、PIMO マップまたはメッセージ・ハンドラー、およびデータ・ハンドラーの 3 つの部分で構成されています。Request Pool から着信するイベント・データは、PIMO フレームワークを介して実行されますが、ここでは、メッセージ・ハンドラーに存在する機能を使用して特定の前処理操作が実行されます。データは、次に、さまざまなアダプターが使用する独立したプラグインであるデータ・ハンドラーに渡され、WBI が使用可能なビジネス・オブジェクトに組み込まれます。データ・ハンドラーの選択は、DataHandlerMimeType と、CFG における関連のプロパティーが基準になります。ビジネス・オブジェクトは、ブローカーに渡されます。ビジネス・オブジェクトの構造は、すべてがデータ・ハンドラーによって決定されるため、TCP/IP アダプターは、関連のデータ・ハンドラーが存在するあらゆる種類のメッセージ・データを処理できます。

サービス呼び出し要求処理は、同じ段階を逆方向にたどります。つまり、データ・ハンドラーがビジネス・オブジェクトを適切なメッセージ構造に変換し、PIMO フレームワークがメッセージ・ハンドラーを使用して後処理を実行します。その後、メッセージは接続マネージャーの Protocol Handler フレームワークに渡され、ここからサービス要求がリモート・ホストに送出されます。

PIMO フレームワーク

TCP/IP 接続を介して着信するメッセージ構造をデータ・ハンドラーに送信して WebSphere Business Integration Server Express ビジネス・オブジェクトに変換するには、その前にメッセージ構造に対して一定の規模の前処理が必要な場合があります。

例えば、HL7 という医療業界のデータ標準について考えます。ネットワーク送信エラーの検出と修正の詳細は、最新のネットワーク・プロトコルの大半では下位レベルが処理するため、主要な標準には、これらを網羅する仕様が盛り込まれていません。ただし、HL7 データ・フローの一部になる場合がある多くのミニ・コンピューター・システムやメインフレーム・コンピューター・システムは、下位層の機能が十分には提供されない通信環境で動作します。これらの場合に、HL7 は、Hybrid Low Layer Protocol や Minimal Low Layer Protocol など、異なる環境に適合するい

くつかの下位層プロトコルを代替として提供します。これらのプロトコルを使用して送信されるメッセージについては、データの本体を抽出する前に前処理して、このプロトコルに関連する情報を削除しておく必要があります。

TCP/IP コネクターの PIMO インフラストラクチャーは、まさにこの種の前処理を実行する目的で設計されています。Production Instruction Meta Object フレームワークは、コネクター・レベルでのビジネス・ロジック処理に効果を発揮するための、柔軟性の高い汎用の抽象概念です。PIMO インフラストラクチャーは、幅広い動作が可能であり、他のアダプター内では別の形で使用されます。TCP/IP コネクターの内部では、これらの前処理問題および後処理問題の対処専用にカスタマイズされています。

PIMO フレームワークは、専用に設計された 1 組のメタオブジェクトを使用して、その作業を実行します。アダプターの PIMO 階層の最上位は、`BIA_MO_Tcpip_MapSubscriptions` オブジェクトです。 15 ページの『BIA_MO_Tcpip_MapSubscriptions』 の図に、このオブジェクトを Business Object Designer Express で表示した場合の外観を示します。このオブジェクトは、データの経路となるインバウンド (イベント前処理) パスおよびアウトバウンド (サービス呼び出し要求後処理) パスの両方を指定します。このオブジェクトには、`BIA_MO_Tcpip_MapSubscriptions_In` および同等の Out オブジェクトの 2 つのオブジェクトが格納されており、それぞれのオブジェクトには、該当する PIMO マップ・オブジェクトへの参照が格納されています。前述の HL7 オブジェクトの場合、該当する `In` PIMO マップ・オブジェクトは、`BIA_Map_InputMessage_to_LLPMessageList` になります。 16 ページの『BIA_Map_InputMessage_to_LLPMessageList』を参照すると、このオブジェクトを Business Object Designer Express で表示した場合の外観が分かります。

この PIMO マップ・オブジェクトは、コア PIMO オブジェクトとして機能します。PIMO オブジェクトは、Port、Declaration、および Action の 3 つの基本属性で構成されます。

各 Port は、さらに IPort と OPort の 2 つの属性で構成されます。これらは、送信元オブジェクト (例えば、イベント処理の内部ラッパー・オブジェクトである BIA_InputMessage オブジェクト) と宛先オブジェクト (BIA_LLPMessage List オブジェクト) の予想タイプを示しています。15 ページの例では、チェーニングされた (連動する) 2 つの入力マップを示します。最初のマップは複数のメッセージを単一メッセージに分離し、2 番目のマップは実際の HL7 メッセージから LLP 情報を除去します。この処理を複数のステップに分割すると、チェーニング・マップは、より複雑なタイプの処理を生成できます。チェーニングされたマップを使用する場合は、ステップ 1 の OPort タイプは、ステップ 2 の IPort タイプと完全に一致するなどの条件が必須です。

Declaration 属性はオプションです。この属性には、処理中に使用する一時変数の名前が書き込まれています。この例では、Declaration オブジェクトに「contentText」などの名前が格納されています。

最後に、PIMO マップには Action 属性が格納されます。各 Action 属性は、定義済みの 1 つ以上の Action で構成されます。定義済みの Action ごとのアプリケーション固有の情報 (ASI) は、必要な実際のデータ変換を実行する目的で設計されたネイティブ Java クラスであるメッセージ・ハンドラーを呼び出すために PIMO が必

要とする情報を提供します。16 ページの ASI の例から、必要な情報を示します。その内容は次のとおりです。

```
type=nativeStatic;
class=com.im.adapters.tcpip.messagehandlers.LLPMessagingProtocoHandler;
method=parseInputMessageToLLPMessages; target=contentText;IPort;Oport
```

ASI には、次の情報が記述されています。

type = このリリースでは、この値は必ず nativeStatic になります。現在の PIMO 構造体がサポートしているのは、共通の静的メソッドのみです。

class =
この値は、メソッドを格納している Java クラスの完全修飾名を表します。この例では、
com.ibm.adapters.tcpip.messagehandlers.LLPMessagingProtocolHandler
です。

method =
この値は、呼び出しの対象となるメソッドを表します。この例では、
parseInputMessageToLLPMessages です。

target =
この値は、戻されたデータの格納場所を表します。この例では、データの格納先は Declaration 属性に作成された変数 contentText です。

var1, var2 . . . varx
メソッドに渡されるパラメーターのオープン・リストです。このリストにおける変数の順序と数は、メソッドが予想するパラメーターの順序と数に完全に一致する必要があります。この例では、var1 と var2 は、それぞれ Port 属性の IPort ビジネス・オブジェクトおよび OPort ビジネス・オブジェクトになります。

この設定の中核をなすメソッドである parseInputMessageToLLPMessages には、LLP 固有のラッパー・データを抽出し、LLPMessage の個々の部分 (ヘッダー、メッセージ自体、およびトレーラー) を格納するリストを戻す方法が組み込まれています。これにより、メッセージを (BIA_ContentBO として) データ・ハンドラーに渡すことができます。これらの一連の HL7 メッセージ・ハンドラーは、TCP/IP のインストール環境に組み込まれていますが、その他も必要に応じて開発できます。

繰り返しますが、サービス呼び出し要求処理は、同じ段階を逆方向にたどります。PIMO フレームワークを使用して、メッセージをそのプロトコル固有データでラップし、その後、リモート・ホストに転送します。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、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 Corporation
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。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。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。著作権使用許諾: 本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

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
CICS
CrossWorlds
DB2
DB2 Universal Database
Lotus
i5/OS
IMS
Informix
iSeries
Lotus Domino
Lotus Notes
MQIntegrator
MQSeries
MVS
OS/400
Passport Advantage
SupportPac
WebSphere
z/OS
Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。
MMX および Pentium は、Intel Corporation の米国およびその他の国における商標です。
Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。
Linux は、Linus Torvalds の米国およびその他の国における商標です。
他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

WebSphere Business Integration Server Express and Express Plus には、Eclipse Project (http://www.eclipse.org/) により開発されたソフトウェアが含まれています。

WebSphere Business Integration Server Express バージョン 4.4 および WebSphere Business Integration Server Express Plus バージョン 4.4

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

M

P

T

U

W

Printed in Japan

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