IBM WebSphere Business Integration Server Express and Express Plus

Adapter for HTTP ユーザーズ・ガイド

アダプター・バージョン 1.2.1

IBM WebSphere Business Integration Server Express and Express Plus

Adapter for HTTP ユーザーズ・ガイド

アダプター・バージョン 1.2.1

お願い

本書および本書で紹介する製品をご使用になる前に、 141 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere Business Integration Adapter for HTTP (5724-H49) バージョン 1.2.1 に適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原 典： WebSphere Business Integration
Server Express and Express Plus
Adapter for HTTP User Guide
Adapter Version 1.2.1

発 行： 日本アイ・ビー・エム株式会社

担 当： ナショナル・ランゲージ・サポート

第1刷 2005.8

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

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

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

対象読者

本書の対象読者は、IBM WebSphere の顧客、コンサルタント、開発者など、Adapter for HTTP をインプリメントする人々です。

本書を読むための前提条件

本書の中では、いろいろな前提条件が挙げられています。これらのうちの多くは、http プロトコルに関する情報またはリソースを含む Web サイトに対する参照から構成されています。また、WebSphere Business Integration システムのインプリメントに関する知識も必要です。最初に「システム・インプリメンテーション・ガイド」を読むことをお勧めします。この資料には、詳細な情報が記載された文献に関する相互参照が示されています。

関連文書

本書の対象製品の一連の関連文書には、WebSphere Business Integration Server Express のどのインストールにも共通する機能とコンポーネントの解説のほか、特定のコンポーネントに関する参考資料が含まれています。

関連文書は、http://www.ibm.com/websphere/wbiserverexpress/infocenter でダウンロード、インストール、および表示することができます。

表記上の規則

本書では、以下のような規則を使用しています。

`Courier フォント`	コマンド名、ファイル名、ユーザーの入力した情報、システムが画面に出力した情報などのリテラル値を示します。

太字	初出語を示します。
イタリック、*イタリック*	変数名または相互参照を示します。
青のアウトライン	オンラインで表示したときにのみ見られる青のアウトラインは、相互参照用のハイパーリンクです。アウトラインの内側をクリックすると、参照先オブジェクトにジャンプします。
{ }	構文の記述行の場合、中括弧 {} で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
[ ]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプショナル・パラメーターです。
...	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、`option[,...]` は、複数のオプションをコンマで区切って指定できることを意味します。
< >	命名規則により、1 つの名前の各エレメントを個々に判別できるようにするために、不等号括弧で囲みます。例えば、`<server_name><connector_name>tmp.log` のように使用します。
/, ¥	本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。i5/OS および 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 IBM 製品パス名は、製品がシステムにインストールされているディレクトリーを基準にした相対パス名です。
ProductDir	製品のインストール先ディレクトリーを表します。各プラットフォームのデフォルトは、以下のとおりです。 Windows: IBM¥WebSphereServer i5/OS: /QIBM/ProdData/WBIServer44/Product Linux: /home/${username}/IBM/WebSphereServer
->	メニューから次のように項目を選択します。「ファイルの選択 (Choose File)」->「更新 (Update)」->「SGML リファレンス (SGML References)」

本リリースの新機能

リリース 1.2.1 の新機能

- WebSphere Business Integration Adapter Framework、バージョン 2.6.0 をサポートします。
- アダプターが非アクティブのときに HTTP(S) リスナーが要求を受け入れないよう停止します。

リリース 1.1 の新機能

このリリースには、以下の機能拡張が含まれます。

- java.protocol.handler.pkgs の値を指定していない場合、コネクターは初期化中にデフォルト値を使用します。詳細については、61 ページの『JSSE』を参照してください。
- HTTP プロトコル・リスナーは、任意の受信ヘッダー値を持つ要求をサポートします。必要な場合は、ヘッダーの検証をコラボレーションに委任できます。
- コネクター固有プロパティー WorkerThreadCount の最小値が変更されました。詳細については、67 ページの『WorkerThreadCount』を参照してください。
- HTTP(S) リスナーによる同期イベント処理の場合に、応答がコラボレーションによって取り込まれないときは、応答の Content-Type HTTP ヘッダーの ContentType 部分が要求の ContentType に設定されます。

第 1 章 アダプターの概要

コネクターは、WebSphere Business Integration Server Express Adapter for HTTP のランタイム・コンポーネントです。コネクターを使用すると、各企業は、自社の組織内部や取引先で使用するための HTTP(S) メッセージを集約、公開、および利用することができます。本書で説明するコネクターおよびその他のコンポーネントは、HTTP および HTTPS プロトコルによって伝達できるメッセージ本体のビジネス・オブジェクト情報の交換に必要な機能を提供します。

この章では、WebSphere Business Integration Server Express Adapter for HTTP のインプリメントに必要なスコープ、コンポーネント、設計ツール、およびアーキテクチャーについて説明します。また、本書で述べられている HTTP コンポーネントをインストールおよび構成するために行う必要のあるタスクについても概説します。コンポーネントのインストールおよび構成については、 9 ページの『インストール、構成、および設計のチェックリスト』を参照してください。

注: Adapter for HTTP は、標準 Adapter Framework API をインプリメントしています。ただし、アダプターが提供する機能は、厳密には InterChange Server Express 統合ブローカーをサポートするように設計されています。

Adapter for HTTP の環境

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

ソフトウェア前提条件

Connector for HTTP をインストールする前に、以下の前提事項およびソフトウェア要件を確認してください。

- HTTPS/SSL を使用する場合は、鍵ストアおよびトラストストアを作成するために、ユーザー独自のサード・パーティー・ソフトウェアが必要になります。

アダプター・プラットフォーム

アダプターは以下のプラットフォーム上で稼働します (オペレーティング・システム)。

- Microsoft Windows 2003
- Linux:
  RedHat Enterprise Linux WS/AS/ES 3.0 Update 2、Intel (IA32)
  SuSE Linux ES 8.1 SP3、Intel (IA32)
  SuSE Linux ES 9.0、Intel (IA32)
- IBM i5/OS V5R3 および OS/400 V5R2

  注: 明示されない限り、i5/OS は OS/400 および i5/OS を意味します。

規格および API

さまざまな規格や技術により、ネットワークを介してほかの Web サービスの機能にアクセスできるようになります。

アダプターが使用する規格は、以下のとおりです。

- HTTP 1.0

アダプターが使用する API は、以下のとおりです。

- IBM JSSE 1.0.2

構成によっては、追加のソフトウェアをインストールする必要がある場合もあります。構成別の要件については、以下のセクションで説明します。

SSL

SSL の使用を計画している場合は、鍵ストア、証明書、および鍵生成を管理するために、サード・パーティーのソフトウェアを使用する必要があります。鍵ストア、証明書のセットアップ、または鍵生成用のツールは提供されていません。Keytool (IBM JRE に同梱) の使用を選択して、自己署名証明書を作成し、鍵ストアを管理することもできます。詳しくは、61 ページの『SSL』を参照してください。

ロケール依存データ

コネクターは、2 バイト文字セットをサポートできるようにグローバル化対応されています。ある文字コードを使用するロケーションから別のコード・セットを使用するロケーションへ、コネクターがデータを転送するとき、コネクターはデータの意味を保存するため、文字変換を実行します。

Java 仮想マシン (JVM) 内の Java ランタイム環境では、データは Unicode 文字コード・セットで表現されます。Unicode には、ほとんどの既知の文字コード・セット (1 バイト系とマルチバイト系をいずれも含む) に対応できるエンコード方式が組み込まれています。WebSphere Business Integration システムのコンポーネントの大部分は Java で作成されています。したがって、ほとんどのインテグレーション・コンポーネント間で、文字変換を行わずにデータを転送できます。

注: コネクターは、国際化対応されていません。このことは、トレースおよびログ・メッセージが変換されないということを意味します。

HTTP コネクター

このセクションでは、グローバリゼーションとコネクターについて説明します。

イベント通知: コネクターは、プラグ可能なプロトコル・リスナーを使用してイベント通知を行います。プロトコル・リスナーは、トランスポートからメッセージを抽出し、メッセージ・メタデータで指定されたデータ・ハンドラーを呼び出します。リスナー処理の詳細については、 49 ページの『HTTP および HTTPS プロトコル・リスナー処理』を参照してください。

要求処理: コネクターは、プラグ可能 HTTP-HTTPS プロトコル・ハンドラー・フレームワークを使用して要求処理を行います。プロトコル・ハンドラーはデータ・ハンドラーを呼び出します。詳細については、 57 ページの『HTTP-HTTPS プロトコル・ハンドラー処理』を参照してください。

データ・ハンドラー

HTTP アダプターは、任意のデータ・ハンドラーを使用して構成できます。データ・ハンドラー構成の概要については、 10 ページの『データ・ハンドラーの構成』を参照してください。

用語

本書では、以下の用語を使用します。

- **ASI (アプリケーション固有情報)** は、特定のアプリケーションまたはテクノロジーに合わせて調整されたコードです。ASI は、ビジネス・オブジェクト定義の属性レベルとビジネス・オブジェクト・レベルの両方のレベルで存在します。
- **ASBO (アプリケーション固有ビジネス・オブジェクト)** ASI を含めることのできるビジネス・オブジェクト。
- **BO (ビジネス・オブジェクト)** ビジネス・エンティティー (Customer など) およびデータへの処置 (作成または更新操作など) を表す属性の集合。IBM WebSphere システムのコンポーネントは、情報を交換したりアクションを起動したりするためにビジネス・オブジェクトを使用します。
- **Content-Type** *タイプ/サブタイプ* およびオプショナル・パラメーターを含む HTTP プロトコル・ヘッダー。例えば、Content-Type 値が `text/xml;charset=ISO-8859-1` の場合、`text/xml` がタイプ/サブタイプであり、`charset=ISO-8859-1` がオプションの Charset パラメーターです。
- **ContentType** Content-Type ヘッダー値の*タイプ/サブタイプ* 部分のみを意味します。例えば、Content-Type 値が `text/xml;charset=ISO-8859-1` の場合、本書では `text/xml` が ContentType と呼ばれます。
- **GBO (汎用ビジネス・オブジェクト)** ASI を含まず、どのアプリケーションとも関連していないビジネス・オブジェクト。
- **MO_DataHandler_Default** コネクター・エージェントが、どのデータ・ハンドラーのインスタンスを生成するかを決定するために使用するデータ・ハンドラー・メタオブジェクト。これは、コネクターの DataHandlerMetaObjectName 構成プロパティーで指定されます。
- **プロトコル構成 MO** 要求処理の際に、HTTP-HTTPS プロトコル・ハンドラーは、プロトコル構成 MO を使用して宛先を判別します。イベント処理中に、コラボレーションを公開する場合、コネクターは、プロトコル構成 MO を使用して、メッセージ・ヘッダー情報を HTTP または HTTPS プロトコル・リスナーからコラボレーションへ転送します。

- **トップレベル・ビジネス・オブジェクト** トップレベル・ビジネス・オブジェクトには、1 つの要求、1 つの応答 (オプション)、および 1 つ以上の障害 (オプション) ビジネス・オブジェクトが含まれます。TLO は、コネクターがイベント処理および要求処理の両方で使用します。

Connector for HTTP のコンポーネント

図 1 は、Connector for HTTP を例示したもので、プロトコル・ハンドラー・フレームワークおよびプロトコル・リスナー・フレームワークが含まれています。

注: Adapter for HTTP には、XML データ・ハンドラーの制限された使用ライセンスが付いてきます。ただし、アダプターの動作には XML データ・ハンドラーは必要ありません。

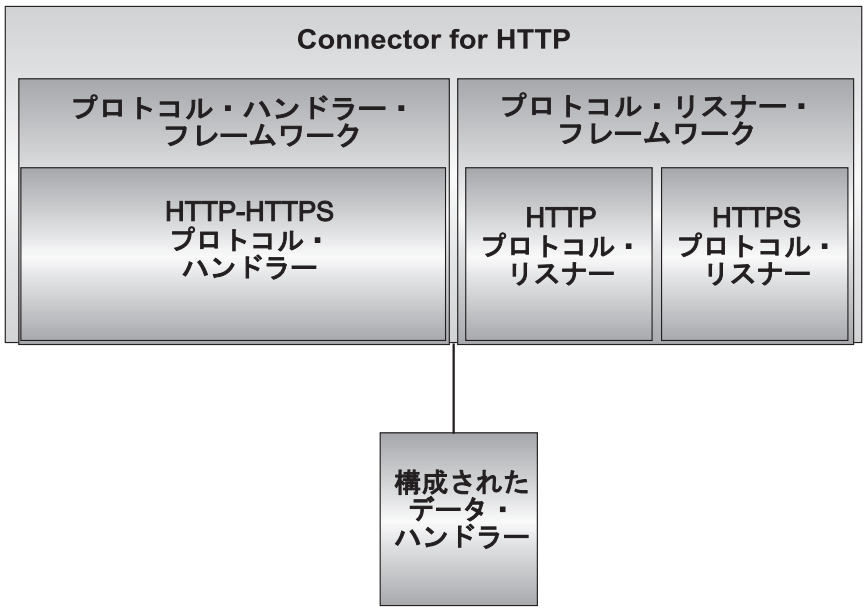


図 1. Connector for HTTP

以下のコンポーネントは、相互に作用することにより、インターネットを介したデータ交換を可能にします。

- 構成されたデータ・ハンドラーおよびプロトコルのリスナーとハンドラーを含む、HTTP コネクター
- HTTP 使用可能コラボレーション
- ビジネス・オブジェクトおよび HTTP(S) メッセージ
- InterChange Server Express

Connector for HTTP

要求処理中に、コネクターは、ビジネス・オブジェクトを要求メッセージに変換し、それらを指定された宛先に転送することによって、コラボレーション・サービス呼び出しに応答します。オプションで (同期要求処理の場合)、コネクターは、応答メッセージを応答ビジネス・オブジェクトに変換し、これらをコラボレーションに戻します。

イベント処理中に、コネクターは、要求メッセージを要求ビジネス・オブジェクトに変換し、それらを処理するためにコラボレーションに引き渡すことによって、クライアントからの要求メッセージを処理します。コネクターは、オプションで応答ビジネス・オブジェクトをコラボレーションから受け取ります。これらの応答ビジネス・オブジェクトは、応答メッセージに変換されてから、クライアントに戻されます。

詳しくは、 45 ページの『第 4 章 HTTP コネクター』を参照してください。

注: この資料でコネクターと言う場合、特に指定のないかぎり、HTTP コネクターのことを指します。

プロトコル・リスナーおよびハンドラー

コネクターには、以下のプロトコル・リスナーおよびハンドラーが組み込まれています。

- HTTP プロトコル・リスナー
- HTTPS プロトコル・リスナー
- HTTP-HTTPS プロトコル・ハンドラー

プロトコル・リスナーは、内部または外部クライアントから、HTTP または HTTPS フォーマットのイベントを検出します。これらは、コネクターに対し、コラボレーションによる処理が必要なイベントを通知します。次に、プロトコル・リスナーは、プロトコル構成オブジェクトに組み込まれたビジネス・オブジェクト・レベルと属性レベルの ASI、コネクター・プロパティー、および変換ルールを読み取り、コラボレーション、データ・ハンドラー、処理モード (同期/非同期)、およびトランザクションのトランスポート固有の特徴を判別します。プロトコル・リスナー処理の詳細な説明については、 48 ページの『プロトコル・リスナー』を参照してください。

プロトコル・ハンドラーは、コラボレーションのために、HTTP または HTTPS フォーマットの HTTP サービスを呼び出します。HTTP(S) プロトコル・ハンドラーは、プロトコル構成オブジェクトに組み込まれた TLO ASI および変換ルールを読み取り、要求 (データ・ハンドラーはこれを使用して、メッセージをビジネス・オブジェクトに変換したり、その逆を行ったりします) をどのように処理 (同期または非同期) するかを判別します。また、要求ビジネス・オブジェクトのプロトコル構成 MO の Destination 属性から宛先も判別します。同期トランザクションの場合、プロトコル・ハンドラーは、応答メッセージを処理して、応答ビジネス・オブジェクトに変換し、コラボレーションに戻します。

プロトコル・ハンドラーについて詳しくは、 56 ページの『プロトコル処理』を参照してください。

データ・ハンドラー

HTTP アダプターは、任意のデータ・ハンドラーを使用して構成できます。図示するために、本書ではしばしば、`text/xml` MIME タイプおよび XML データ・ハンドラーを参照します。

構成されたデータ・ハンドラーは、ビジネス・オブジェクトからメッセージへの変換およびその逆の変換を行います。詳細については、HTTP アダプターで使用しているデータ・ハンドラーの資料を参照してください。

Object Discovery Agents

オブジェクト・ディスカバリー・エージェント (ODA) があるデータ・ハンドラーを使用している場合、その ODA を使用してビジネス・オブジェクトを生成できます。例えば、要件に XML エンコード方式が含まれており、XML データ・ハンドラーを使用してアダプターを構成した場合、XML ODA を使用してビジネス・オブジェクトを作成し、変更することができます。

コネクターの配置

HTTP コネクターを配置するには、以下の 2 つの方法があります。

- ファイアウォールの内側で、ビジネス・プロセス同士が HTTP または HTTPS フォーマットで通信しているエンタープライズ内のイントラネット・ベースのソリューションとして (図 2 を参照)。

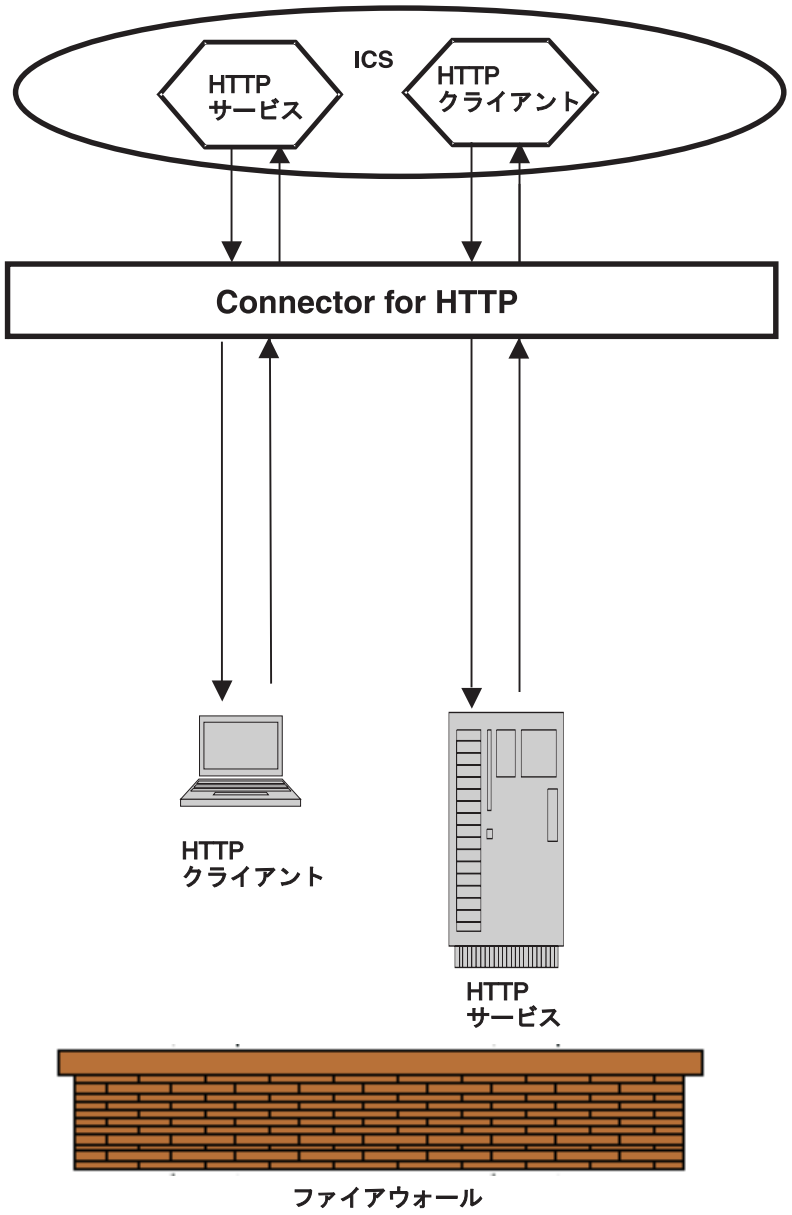


図 2. イントラネット・ソリューションとしての HTTP アダプター

- ファイアウォールの内側で、エンタープライズの外部との通信を、処理、フィルター操作、または管理するフロントエンドまたはゲートウェイ・サーバーと連携して。

注: HTTP コネクターには、外部からの着信メッセージ、または外部への発信メッセージを管理するためのゲートウェイまたはフロントエンドが組み込まれていません。ユーザー独自のゲートウェイを構成および配置する必要があります。***コネクターは、DMZ 内またはファイアウォールの外側ではなく、エンタープライズ内でのみ配置する必要があります。***

Connector for HTTP のアーキテクチャー

上位レベルのコンポーネントのアーキテクチャーを示すため、このセクションでは、2 つのデータ・フローについて説明します。図 3 に 2 つのシナリオを示します。これら 2 つのコンポーネントについて、以下で説明します。

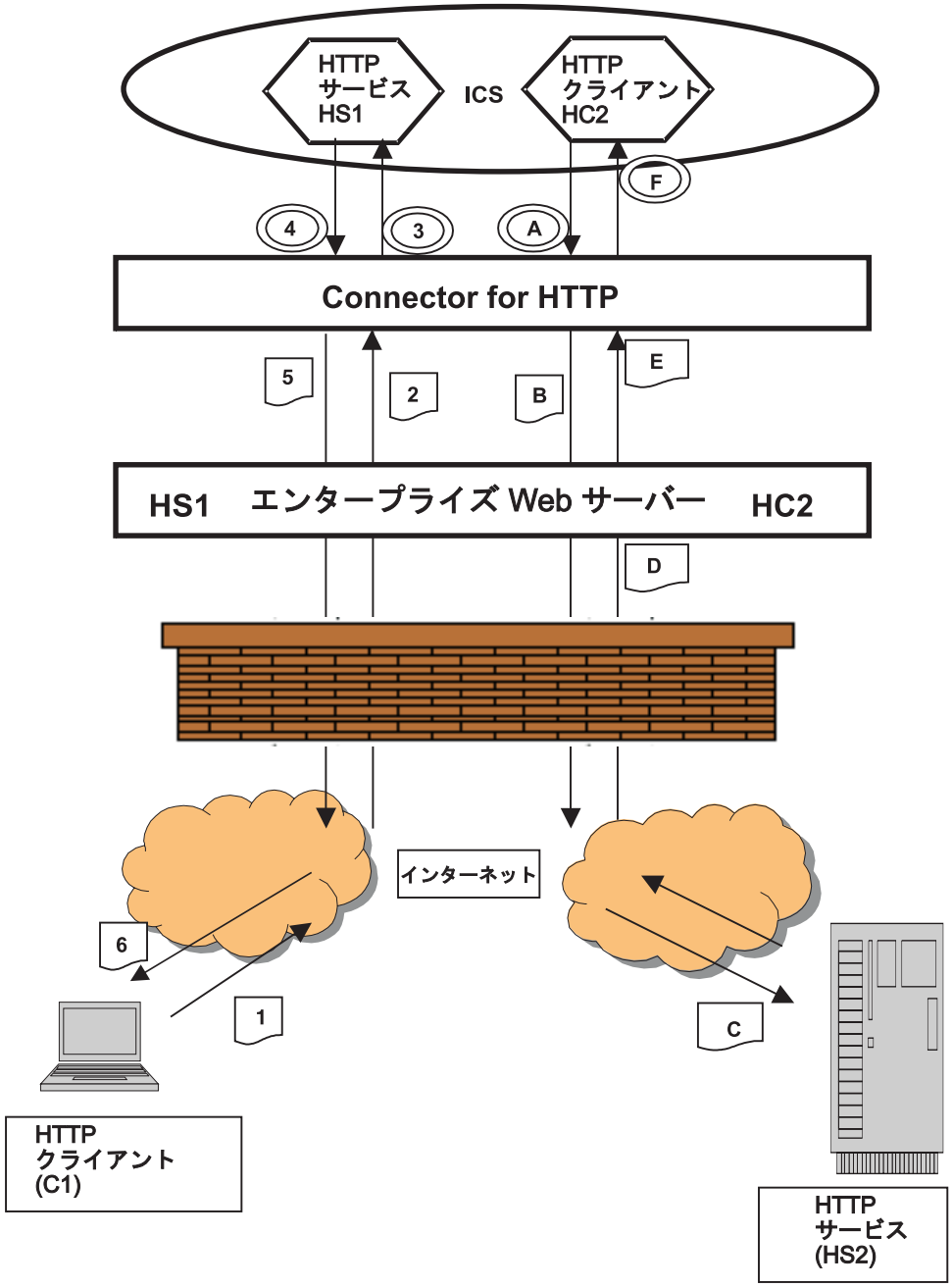


図 3. HTTP メッセージの流れ

要求処理は、コラボレーションがコネクターに対してサービス呼び出し要求を実行したときに発生するイベントの順序を表しています。このシナリオでは、コラボレーションの果たす役割はクライアントであり、要求をサーバーに送信しています。

A　コラボレーションはサービス呼び出し要求をコネクターに送信し、コネクターはデータ・ハンドラーを呼び出してビジネス・オブジェクトを要求メッセージに変換します。

B　コネクターは、要求メッセージを送信することによって、エンタープライズ Web サーバーの URL を呼び出します。

C　エンタープライズ Web サーバーは、HTTP サーバー (HS2) の URL を呼び出します。

D　HTTP サーバー HS2 は要求を処理し、応答を戻します。応答は、同一接続の一部として戻されます。

E　エンタープライズ Web サーバーは、応答メッセージをアダプターに戻します。

F　コネクターは応答 (または障害) メッセージを受け取り、データ・ハンドラーを呼び出してメッセージをビジネス・オブジェクトに変換し、それをコラボレーションに戻します。

イベント処理は、コラボレーションが HTTP クライアントに呼び出されたときに発生するイベントの順序を表しています。このシナリオでは、コラボレーションはサーバーの役割を果たし、(外部または内部) クライアントからの要求を受け入れ、必要に応じて応答を行います。

1　HTTP クライアント (C1) は、要求メッセージを宛先のコラボレーションに送信します。

2　HTTP クライアント が外部の場合、ゲートウェイはメッセージを受信してコネクターに発送します。

3　コネクターは、ビジネス・オブジェクトに変換するために、メッセージをデータ・ハンドラーに送信します。コネクターはコラボレーションを呼び出します。

4　コラボレーションは、応答 (または障害) ビジネス・オブジェクトを戻します。

5　コネクターは、データ・ハンドラーを呼び出して、応答 (または障害) ビジネス・オブジェクトを応答メッセージに変換します。コネクターは、応答をゲートウェイに戻します。

6　クライアントが外部の場合、ゲートウェイは応答メッセージを HTTP クライアント (C1) に発送します。

インストール、構成、および設計のチェックリスト

このセクションでは、HTTP ソリューションをインストール、構成、および設計するときに行う必要のある作業を要約します。各セクションでは、作業を簡単に説明したうえで、その作業の実行方法や背景情報を説明する、本書中の該当セクションへのリンク (および関連資料への相互参照) を提供します。

アダプターのインストール

何をどこにインストールする必要があるのかについては、11 ページの『第 2 章 インストールおよび始動』を参照してください。

コネクター・プロパティーの構成

コネクターには、2 種類の構成プロパティーがあります。標準の構成プロパティーと、コネクター固有の構成プロパティーです。一部のプロパティーはデフォルト値を持っており、変更を加えなくても使用できます。また、一部のプロパティーについては、コネクターを実行する前に値を設定する必要があります。詳しくは、45 ページの『第 4 章 HTTP コネクター』を参照してください。

プロトコル・ハンドラーおよびプロトコル・リスナーの構成

プロトコル・ハンドラーおよびプロトコル・リスナーの構成は、これらのコンポーネントの振る舞いを制御するコネクター構成プロパティーに値を割り当てるときに行います。詳しくは、45 ページの『第 4 章 HTTP コネクター』を参照してください。

ビジネス・オブジェクトの作成および変更

HTTP コネクターで使用するデータ・ハンドラーによって、ODA を使用できる場合があります。 ODA は、ビジネス・オブジェクトの作成と変更のプロセスを自動化します。そうでない場合、Business Object Designer Express を使用して、ビジネス・オブジェクトの作成と変更を手動で行うことができます。詳細については、使用しているデータ・ハンドラーの資料および「*ビジネス・オブジェクト開発ガイド*」を参照してください。

データ・ハンドラーの構成

製品ファイルをインストールしたら、始動する前に、データ・ハンドラー・メタオブジェクトを構成してください。コネクター固有構成プロパティーの DataHandlerMetaObjectName を指定して始動します。データ・ハンドラーが構成プロパティーの検索に使用するトップレベルのメタオブジェクト (MO_DataHandler_Default) の名前を指定します。次に、使用するデータ・ハンドラーに必要なすべての追加構成ステップに従います。オプションとして、MimeType TLO 属性を使用してデータ・ハンドラーを指定できます。詳細については、22 ページの表 6 を参照してください。

データ・ハンドラーの構成に関する詳細については、64 ページの『コネクター固有の構成プロパティー』を参照してください。

第 2 章 インストールおよび始動

- 『インストール作業の概要』
- 『コネクターと関連ファイルのインストール』
- 13 ページの『構成作業の概要』
- 14 ページの『アダプターの複数インスタンスの実行』
- 16 ページの『コネクターの始動』
- 18 ページの『コネクターの停止』

この章では、Connector for HTTP をインプリメントするためのコンポーネントのインストール方法について説明します。InterChange Server Express システムのインストールについての情報は、「*WebSphere Business Integration Server Express インストール・ガイド (Windows 版、Linux 版、または OS/400 および i5/OS 版)*」を参照してください。

インストール作業の概要

ブローカーの互換性、アダプター・フレームワーク、ソフトウェア前提条件、依存関係、および標準と API については、 1 ページの『Adapter for HTTP の環境』 を参照してください。

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

InterChange Server Express のインストール

この作業にはシステムのインストールと InterChange Server Express の始動が含まれますが、これについては、「*システム・インプリメンテーション・ガイド*」に記述されています。InterChange Server Express は、バージョン 4.4 をインストールする必要があります。

リポジトリーにファイルをロードするには、「*システム・インプリメンテーション・ガイド*」を参照してください。

コネクターおよび関連ファイルのインストール

この作業の内容は、ソフトウェア・パッケージからコネクター (および関連コンポーネント) のファイルを使用システムにインストールすることです。『コネクターと関連ファイルのインストール』を参照してください。

コネクターと関連ファイルのインストール

WebSphere Business Integration Server Express アダプター製品のインストールの詳細については、「WebSphere Business Integration Server Express インストール・ガイド」(Windows 版、Linux 版、OS/400 および i5/OS 版) を参照してください。このガイドは、WebSphere Business Integration Server Express Adapters Infocenter (http://www.ibm.com/websphere/wbiserverexpress/infocenter) にあります。

インストール済みファイルの構造

このセクションの表では、インストール済みファイルの構造を示します。

Windows のコネクター・ファイル構造

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

ユーティリティーにより、コネクターがインストールされ、コネクター・エージェントのショートカットが「スタート」メニューに追加されます。

表 1 には、コネクターが使用する Windows ファイルの構造が説明されており、インストーラーによるコネクターのインストールを選択したときに自動的にインストールされるファイルが示されています。

表 1. アダプターにインストールされる Windows ファイルの構造

***ProductDir* のサブディレクトリー**	**説明**
connectors¥HTTP¥BIA_HTTP.jar	WebSphere Business Integration Server Express Adapter JAR ファイル
connectors¥HTTP¥start_HTTP.bat	コネクターの始動ファイル
connectors¥HTTP¥start_HTTP_service.bat	コネクター・サービスの始動スクリプト
bin¥Data¥App¥HTTPConnectorTemplate	HTTP コネクターのテンプレート
connectors¥HTTP¥dependencies¥ibmjsse.jar	IBM 製の JSSE (Java Secure Socket Extension) API
connectors¥HTTP¥dependencies¥mail.jar	Java Mail API
connectors¥HTTP¥README.htm	使用権
connectors¥messages¥HTTPConnector.txt	コネクター・メッセージ・ファイル

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

Linux のコネクター・ファイル構造

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

表 2 には、コネクターが使用する Linux ファイルの構造が説明されており、インストーラーによるコネクターのインストールを選択したときに自動的にインストールされるファイルが示されています。

表 2. アダプター用としてインストールされた Linux ファイル構造

***ProductDir* のサブディレクトリー**	**説明**
connectors/HTTP/BIA_HTTP.jar	WebSphere Business Server Express Integration Adapter JAR ファイル
connectors/HTTP/start_HTTP.sh	コネクターの始動ファイル
bin/Data/App/HTTPConnectorTemplate	HTTP コネクターのテンプレート
connectors/HTTP/dependencies/ibmjsse.jar	IBM 製の JSSE (Java Secure Socket Extension) API
connectors/HTTP/README.htm	使用権
connectors/HTTP/dependecies/mail.jar	Java Mail API
connectors/messages/HTTPConnector.txt	コネクター・メッセージ・ファイル

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

i5/OS のコネクター・ファイル構造

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

表 2 には、コネクターが使用する i5/OS ファイルの構造が説明されており、インストーラーによるコネクターのインストールを選択したときに自動的にインストールされるファイルが示されています。

表 3. アダプターにインストールされる i5/OS ファイルの構造

***ProductDir* のサブディレクトリー**	**説明**
connectors/HTTP/BIA_HTTP.jar	WebSphere Business Server Express Integration Adapter JAR ファイル
connectors/HTTP/start_HTTP.sh	コネクターの始動ファイル
connectors/HTTP/dependencies/ibmjsse.jar	IBM 製の JSSE (Java Secure Socket Extension) API
connectors/HTTP/dependencies/mail.jar	メール .jar ファイル
connectors/HTTP/README.htm	使用権
connectors/messages/HTTPConnector.txt	コネクター・メッセージ・ファイル

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

構成作業の概要

インストールしたら、始動する前に、コンポーネントを以下のように構成する必要があります。

コネクターの構成

この作業の内容は、コネクターのセットアップと構成です。 63 ページの『コネクターの構成』を参照してください。

ビジネス・オブジェクトの構成

ビジネス・オブジェクトの構成手順は、製品スイートのインプリメント方法をどのように選択するかによって異なります。

- **要求処理** 以下の内容に対応するビジネス・オブジェクトを作成する必要があります。
  - 発信要求メッセージ
  - 要求に対して考えられる応答 (障害の場合も含む)

  詳細については、 19 ページの『第 3 章 ビジネス・オブジェクトの要件』を参照してください。
- **イベント処理** TLO ビジネス・オブジェクトを使用します。詳細については、 19 ページの『第 3 章 ビジネス・オブジェクトの要件』を参照してください。

Business Object Designer Express を使用して手動でビジネス・オブジェクトを作成できます。または、ご使用のデータ・ハンドラーによっては、ビジネス・オブジェクトの作成プロセスを自動化する ODA を使用することもできます。詳細については、ご使用のデータ・ハンドラーの資料を参照してください。

データ・ハンドラーの構成

コネクター固有構成プロパティーの DataHandlerMetaObjectName を指定して、データ・ハンドラーを構成します。データ・ハンドラーが構成プロパティーの検索に使用するトップレベルのメタオブジェクト (MO_DataHandler_Default) の名前を指定します。次に、使用するデータ・ハンドラーに必要なすべての追加構成ステップに従います。

オプションとして、MimeType TLO 属性を使用してデータ・ハンドラーを指定できます。詳細については、 22 ページの表 6 を参照してください。

データ・ハンドラーの構成に関する詳細については、 64 ページの『コネクター固有の構成プロパティー』を参照してください。

コラボレーションの構成

要求処理またはイベント処理にコラボレーションを使用できるようにするには、以下の資料を参照してください。

- *コラボレーション開発ガイド*
- *マップ開発ガイド*

アダプターの複数インスタンスの実行

コネクターの複数インスタンスの作成は、多くの点でカスタム・コネクターの作成と似ています。以下に示すステップを実行することによって、コネクターの複数のインスタンスを作成して実行するように、ご使用のシステムを設定することができます。それには、以下の作業を行う必要があります。

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

新規ディレクトリーの作成

- **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は、コネクター・インスタンス用のメタオブジェクトを作成する必要があります。メタオブジェクトをファイルとして保管する場合は、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 サーバー名は、例えば start_HTTP.bat connName serverName のように、startup.bat のパラメーターとして指定できます。

- **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、WebSphereICSName はコネクターの実行に使用する 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は、コネクター・インスタンス用のメタオブジェクトを作成する必要があります。メタオブジェクトをファイルとして保管する場合は、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。

  /QIBM/UserData/WBIServer44/WebSphereICSName/repository/connectorInstance。ここで WebSphereICSName はコネクターの実行に使用する 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は、コネクター・インスタンス用のメタオブジェクトを作成する必要があります。メタオブジェクトをファイルとして保管する場合は、ディレクトリー ProductDir/repository/connectorInstance を作成し、ファイルをここに格納します。InterChange Server Express のサーバー名を connector_manager のパラメーターとして指定することができます。例: connector_manager -start connName WebSphereICSName [-cConfigFile]

ビジネス・オブジェクト定義の作成

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

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

   ProductDir¥repository¥initialConnectorInstance

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

コネクター定義の作成

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

1. 初期コネクターの構成ファイル (コネクター定義) をコピーし、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、 15 ページの『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 の場合のみ) 始動スクリプトのショートカットを作成します。
4. (Windows の場合のみ) 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 の場合のみ) 次の情報を使用して、コネクターのジョブ記述を作成します。
   CRTDUPOBJ(QWBIHTTP) FROMLIB(QWBISVR44)OBJTYPE(*JOBD)TOLIB (QWBISVR44) NEWOBJ(newhttpname)。ここで、newhttpname は新規コネクターのジョブ記述で使用する 10 文字の名前です。
6. (i5/OS の場合のみ) 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、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。

コネクターの始動

コネクターは、**コネクター始動スクリプト**を使用して明示的に開始する必要があります。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¥*connName* に存在していなければなりません。ここで、*connName* はコネクターを示します。

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する /QIBM/UserData/WBIServer44/<instance>/connectors/<ConnInstance/ に存在しなければなりません。

表 4 が示すとおり、始動スクリプトの名前はオペレーティング・システム・プラットフォームにより異なります。

表 4. コネクターの始動スクリプト

オペレーティング・システム	始動スクリプト
Linux	connector_manager
i5/OS	start_connName.sh
Windows	start_*connName*.bat

始動スクリプトが実行されると、デフォルトで構成ファイルは *Productdir* にあることが要求されます (下記のコマンドを参照)。ここに構成ファイルを格納します。

注: アダプターが JMS トランスポートを使用している場合は、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開始**:
  - 「**スタート**」メニューから、「**プログラム**」>「**IBM WebSphere Business Integration Server 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。ただし、これはカスタマイズすることができます。または、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、次を入力します: start_connName connName brokerName {-cconfigFile}
  - Windows システムでは、コネクターを Windows のサービスとして開始されるように構成できる。この場合、自動サービスでは Windows システムがブートするとき、手動サービスでは「Windows サービス」ウィンドウからサービスを開始するときに、コネクターが開始されます。
- **Linux システムでのコネクターの開始**:
  - コマンド行から、以下を入力します。

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

    ここで、*connName* はコネクターの名前であり、*brokerName* はご使用の統合ブローカーを表します。
  - 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開始**:
  - WebSphere Business Integrations Server Express Console 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、OS/400 または i5/OS システム名または IP アドレスと、*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て、「開始」をクリックします。
  - コンソールを使用してアダプターを自動的に開始するには、submit_adapter.sh スクリプトを使用します。これが、サーバーの自動開始ジョブ・エントリー内のサブシステムを使用してアダプターが開始する唯一の方法です。
  - バッチ・モードでは、i5/OS コマンド行から CL コマンド QSH を実行し、QSHELL 環境から /QIBM/ProdData/WBIServer44/bin/submit_adapter.sh *connName WebSphereICSName pathToConnNameStartScript jobDescriptionName* を実行する必要があります。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。
  - 対話モードでは、CL コマンド QSH を実行し、QSHELL 環境から */QIBM/UserData/WBIServer44/WebSphereICSName/connectors/connName/start_connName.sh connNameWebsphereICSName* [-cConfigFile] を実行する必要があります。ここで、*connName* はコネクターの名前であり、*WebSphereICSName* は InterChange Server Express インスタンスの名前です。

コ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**:
  - コネクター用の別個の「コンソール」ウィンドウを作成する始動スクリプトを起動できます。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。
  - コネクターを Windows のサービスとして始動するように構成できます。この場合、Windows システムがシャットダウンされるとコネクターは停止します。
- **Linux**:

  コネクターはバックグラウンドで実行されるので、個別のウィンドウはありません。その代わりに、以下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* はコネクター名です。
- **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は、次の 2 つの方法のうちの 1 つを使用してコネクターを停止できます。
  - WebSphere Business Integration Server Express Console がインストールされている Windows システムから、「**IBM WebSphere Business Integration Express**」>「**Toolset Express**」>「**管理**」>「**コンソール**」を選択します。次に、OS/400 または i5/OS システム名または IP アドレスと、*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 HTTP アダプターを選択して、「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 製品に対するジョブを表示します。リストをスクロールして、コネクターのジョブ記述に一致するジョブ名を持つジョブを探します。例えば、HTTP コネクターの場合のジョブ名は QWBIHTTPC 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・パラメーターとして *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。
  - QSHELL から start_connName.sh スクリプトを使用してアダプターを始動した場合は、F3 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スクリプト *stop_adapter.sh* を使用して、エージェントを停止することもできます。

第 3 章 ビジネス・オブジェクトの要件

- 『ビジネス・オブジェクトのメタデータ』
- 『コネクター・ビジネス・オブジェクトの構造』
- 20 ページの『同期イベント処理 TLO』
- 28 ページの『非同期イベント処理 TLO』
- 31 ページの『同期要求処理 TLO』
- 31 ページの『同期要求処理 TLO』
- 40 ページの『非同期要求処理 TLO』
- 43 ページの『ビジネス・オブジェクトの開発』

この章では、コネクター・ビジネス・オブジェクトの構造、要件、および属性について説明します。

ビジネス・オブジェクトのメタデータ

Connector for HTTP は、メタデータ主導型のコネクターです。ビジネス・オブジェクトでは、メタデータはアプリケーションに関するデータのことです。このデータはビジネス・オブジェクト定義に格納されており、コネクターとアプリケーションとの対話に役立ちます。メタデータ主導型のコネクターは、コネクター内にハードコーディングされている命令ではなく、ビジネス・オブジェクト定義内にエンコードされているメタデータに基づいて、コネクター自身がサポートしている各ビジネス・オブジェクトを処理します。

ビジネス・オブジェクトのメタデータには、ビジネス・オブジェクトの構造、その属性プロパティーの設定値、およびそのアプリケーション固有情報の内容が含まれています。コネクターは、メタデータ主導型なので、新規や変更後のビジネス・オブジェクトを処理する場合にコネクター・コードを変更する必要がありません。ただし、コネクターの構成済みデータ・ハンドラーでは、そのビジネス・オブジェクトの構造、オブジェクトの基数、アプリケーション固有のテキストのフォーマット、およびビジネス・オブジェクトのデータベース表現について前提事項が存在します。したがって、http のビジネス・オブジェクトを作成または変更する場合、変更の内容はコネクターに対して定められている規則に準拠している必要があります。準拠していない場合、コネクターは新規ビジネス・オブジェクトや変更されたビジネス・オブジェクトを正しく処理できません。

コネクター・ビジネス・オブジェクトの構造

コネクターは、以下のビジネス・オブジェクトを処理できます。

- **TLO** トップレベル・ビジネス・オブジェクト (TLO) には、要求ビジネス・オブジェクトと、オプションとして応答ビジネス・オブジェクトおよび障害ビジネス・オブジェクトが含まれます。これらの子オブジェクトには、内容データおよび、オプションでプロトコル構成 MO も含まれています。それらはデータ・ハンドラー固有オブジェクトでもあります。例えば、XML データ・ハンドラーを使用している場合、子要求は、XML データ・ハンドラーが理解できるビジネス・

オブジェクトです。 TLO、要求オブジェクト、応答オブジェクト、および障害オブジェクトのほかに、アプリケーション固有情報、属性、および要求とイベント処理に関する要件について、以下のセクションで説明および図解します。

注: *TLO は、要求処理およびイベント処理のために使用されます。*

同期イベント処理 TLO

イベント処理のために、コネクターでは、同期および非同期の 2 種類の TLO を使用することが可能です。このセクションでは、同期イベント処理 TLO について説明します。

図 4 は、同期イベント処理のためのビジネス・オブジェクト階層を示しています。要求オブジェクトおよび応答オブジェクトは必須であり、障害オブジェクトはオプションです。

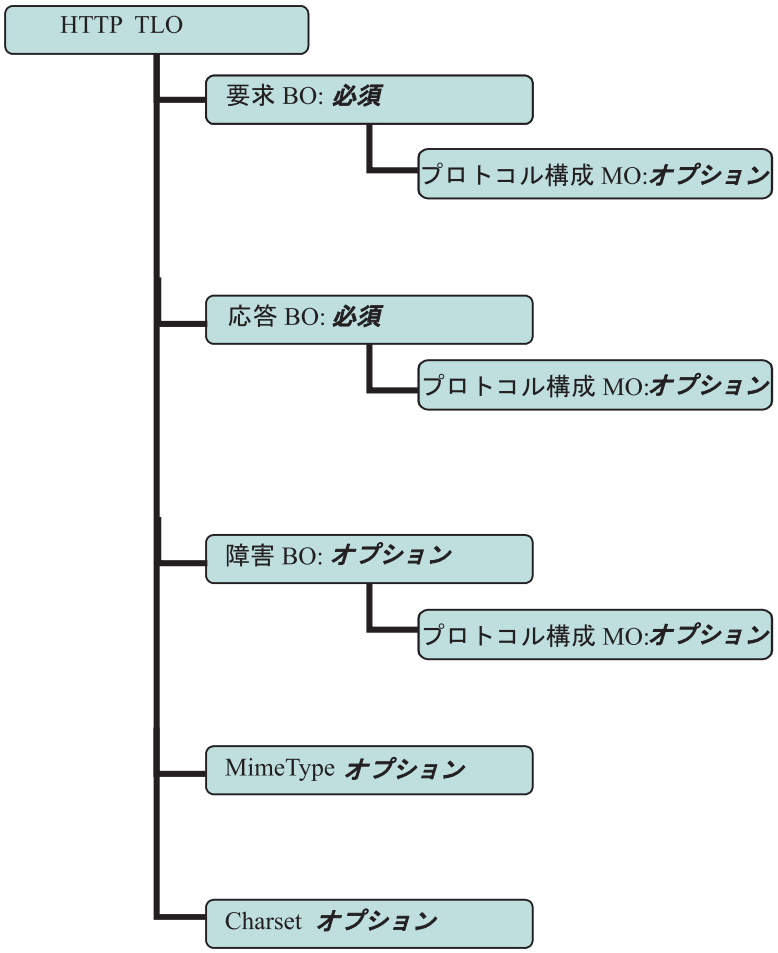


図 4. 同期イベント処理のためのビジネス・オブジェクト階層

TLO には、オブジェクト・レベルの ASI のほか、属性レベルの ASI を持った属性が含まれています。両方の種類の ASI について、以下で説明します。

同期イベント処理 TLO のオブジェクト・レベルの ASI

オブジェクト・レベルの ASI は、TLO の性質、および TLO に含まれるオブジェクトについての基本的情報を提供します。図 5 は、同期イベント処理のためのサンプル TLO である SERVICE_SYNCH_OrderStatus のオブジェクト・レベルの ASI を表しています。

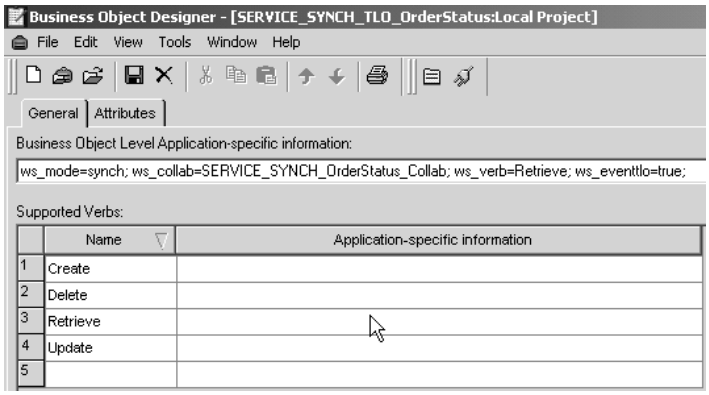


図 5. 同期イベント処理のためのトップレベル・ビジネス・オブジェクト

以下の表 5 では、同期イベント処理 TLO のためのオブジェクト・レベルの ASI について説明します。

表 5. 同期イベント処理 TLO のオブジェクト ASI

オブジェクト・レベルの ASI	説明
`ws_eventtlo=true`	この ASI プロパティーが `true` に設定されている場合、コネクターはこのオブジェクトをイベント処理に使用できる TLO として扱います。
`ws_collab=`*`collabname`*	この ASI は、どのコラボレーションを呼び出すのかをコネクターに指示します。この値はコラボレーションの名前です。 図 5 に示されているサンプルでは、コラボレーション名は SERVICE_SYNCH_OrderStatus_Collab です。
`ws_verb=`*`verb`*	コネクターは、TLO をコラボレーションに引き渡す前に、TLO で動詞を設定するためにこの ASI を使用します。図 5 に示されているサンプルでは、動詞は Retrieve です。
`ws_mode=synch`	イベント通知の際に、コネクターは、この ASI プロパティーを使用して、コラボレーションを同期 (`synch`) で呼び出すのか非同期 (`asynch`) で呼び出すのかを決定します。同期処理の場合は、この ASI を `synch` に設定する必要があります。 デフォルトは `asynch` です。

同期イベント処理 TLO の属性レベルの ASI

それぞれの同期イベント処理 TLO は、属性および属性レベルの ASI を保有しています。図 6 は、サンプル TLO である SERVICE_SYNCH_OrderStatus の属性を表し

ています。また、この図には、属性レベルの ASI が「アプリケーション固有の情報 (App Spec Info)」列に表示されています。

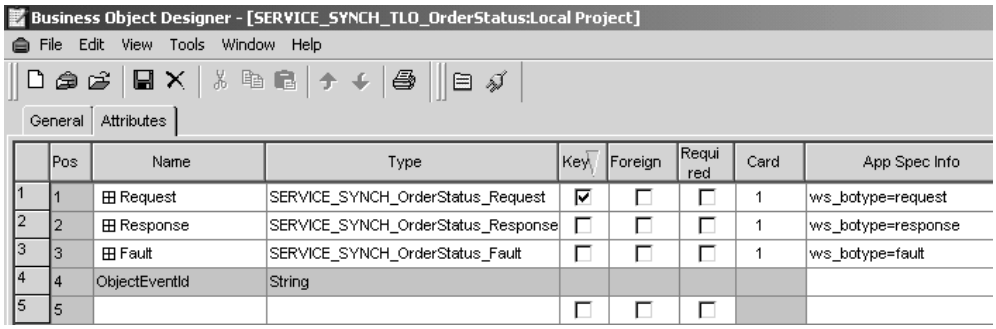


図 6. 同期イベント処理のための TLO 属性

表 6 は、同期イベント処理 TLO の要求、応答、障害、MimeType、Charset の各属性に対する属性レベルの ASI を要約したものです。

表 6. 同期イベント処理 TLO の属性 ASI

TLO 属性	**属性レベルの ASI**	**説明**
MimeType		オプションの属性; 指定されると、その値は、同期応答を呼び出すデータ・ハンドラーの MIME タイプとして使用されます。
Charset		String 型のこのオプショナル・パラメーターは、発信ビジネス・オブジェクトをメッセージに変換するときに、データ・ハンドラーに設定される文字セットを指定します。注: この属性に指定される文字セット値は、応答メッセージの Content-Type プロトコル・ヘッダーには伝搬しません。
Request	`ws_botype=request`	この属性は、HTTP サービス要求に対応します。コネクターはこの ASI を使用して、この TLO 属性のタイプが要求 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。複数の要求属性がある場合は、コネクターは最初の要求属性の ASI を使用します。 この属性は、同期イベント処理 TLO の場合には必須です。

表 6. 同期イベント処理 TLO の属性 ASI (続き)

TLO 属性	属性レベルの ASI	説明
Response	`ws_botype=response`	この属性は、HTTP サービスによって戻される応答に対応します。コネクターはこの ASI を使用して、この TLO 属性のタイプが応答 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。複数の応答属性がある場合は、コネクターは最初の応答属性の ASI を使用します。 この属性は、同期イベント処理 TLO の場合には必須です。
Fault	`ws_botype=fault` `ws_botype=defaultfault`	この属性は、同期イベント処理の場合のオプションであり、応答を正常に取り込むことができないときにコラボレーションから戻される障害メッセージに対応します。コネクターは属性名ではなく、この ASI を使用して、属性タイプが障害 BO であるかどうかを判別します。

同期イベント処理のための要求ビジネス・オブジェクト

要求ビジネス・オブジェクトは、TLO の子であり、同期イベント処理の場合には必須です。要求ビジネス・オブジェクトは、オブジェクト・レベルの ASI を保有しています。同期イベント処理のための要求ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 7 に説明されています。要求ビジネス・オブジェクトに対して、デフォルトの動詞を指定できます。これを行うには、トップレベルの要求ビジネス・オブジェクトの「サポートされている動詞 (Supported Verbs)」リストにある動詞の ASI フィールドに、

```
DefaultVerb=true;
```

と指定します。`DefaultVerb` ASI が指定されず、データ・ハンドラーが動詞の設定されていないビジネス・オブジェクトを処理する場合、そのビジネス・オブジェクトは、動詞なしで戻されます。

表 7. 同期イベント処理: 要求ビジネス・オブジェクトのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
cw_mo_http=*HTTPCfgMO*	この ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。この ASI は、HTTP または HTTPS プロトコル・リスナーを指定します。ASI およびプロトコル構成 MO は、両方ともオプションです。詳しくは、 25 ページの『プロトコル構成 MO』を参照してください。注: ビジネス・オブジェクト変換用に構成するデータ・ハンドラーは、cw_mo で始まる ASI を、変換するビジネス・データの一部としてではなく、メタデータとして読み取ることができる必要があります。XML データ・ハンドラーには、cw_mo メタデータを検出し、その値が指示する属性を無視する機能があります。
ws_tloname=*tloname*	この ASI は、このオブジェクトが属する TLO の名前を指定します。イベント処理の際に、コネクターは、この ASI を使用して、データ・ハンドラーによって引き渡された要求ビジネス・オブジェクトが TLO の子であるかどうかを判別します。引き渡された要求ビジネス・オブジェクトが TLO の子である場合は、コネクターは、指定された TLO を作成し、要求ビジネス・オブジェクトをその TLO の子として設定し、TLO のオブジェクト・レベルの ASI を使用して、この要求ビジネス・オブジェクトを、サブスクライブしているコラボレーションに引き渡します。

同期イベント処理のための応答ビジネス・オブジェクト

応答ビジネス・オブジェクトは、TLO の子であり、同期イベント処理の場合には必須です。同期イベント処理のための応答ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 8 に説明されています。

表 8. 同期イベント処理: 応答ビジネス・オブジェクトのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
cw_mo_http=*HTTPCfgMO*	この ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。この ASI は、HTTP または HTTPS プロトコル・リスナーを指定します。ASI およびプロトコル構成 MO は、両方ともオプションです。詳しくは、 25 ページの『プロトコル構成 MO』を参照してください。注: ビジネス・オブジェクト変換用に構成するデータ・ハンドラーは、cw_mo で始まる ASI を、変換するビジネス・データの一部としてではなく、メタデータとして読み取ることができる必要があります。XML データ・ハンドラーには、cw_mo メタデータを検出し、その値が指示する属性を無視する機能があります。

注: オプションとして、応答 BO のプロトコル構成 MO オブジェクト・レベル ASI を組み込むことができます。

同期イベント処理のための障害ビジネス・オブジェクト

障害ビジネス・オブジェクトは、TLO の子であり、同期イベント処理の場合にはオプションです。同期イベント処理のための障害ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 9 に説明されています。

表 9. 同期イベント処理: 障害ビジネス・オブジェクトのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
cw_mo_http=*HTTPCfgMO*	この ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。この ASI は、HTTP または HTTPS プロトコル・リスナーを指定します。ASI およびプロトコル構成 MO は、両方ともオプションです。詳しくは、『プロトコル構成 MO』を参照してください。注: ビジネス・オブジェクト変換用に構成するデータ・ハンドラーは、cw_mo で始まる ASI を、変換するビジネス・データの一部としてではなく、メタデータとして読み取ることができる必要があります。XML データ・ハンドラーには、cw_mo メタデータを検出し、その値が指示する属性を無視する機能があります。

注: オプションとして、障害 BO のプロトコル構成 MO オブジェクト・レベル ASI を組み込むことができます。

プロトコル構成 MO

プロトコル構成 MO は、オプションとして、イベント処理用の要求、応答、または障害の各ビジネス・オブジェクトの子として組み込まれます。通常、プロトコル・ヘッダーおよびカスタム・プロパティーを (要求メッセージから) 読み取ったり、(応答または障害メッセージへ) 伝搬させたりする必要があるときに指定します。 前述のとおり、要求ビジネス・オブジェクトは、オプションでプロトコル構成 MO の名前をビジネス・オブジェクト・レベルの ASI として宣言します。

- cw_mo_http=*HTTPProtocolListenerConfigMOAttribute*

イベント処理中に、コネクターはプロトコル・リスナー (HTTP または HTTPS) を使用してトランスポートからイベントを検索します。これらのイベントは、コラボレーションのサービスを要求する内部または外部のクライアントからのメッセージです。それぞれのトランスポートには、固有のヘッダー要件があります。コネクターは、プロトコル構成 MO を使用して、プロトコル固有のヘッダー情報をプロトコル・リスナーからコラボレーションへと転送します。プロトコル構成 MO の属性は、インバウンド・メッセージ内のヘッダーに対応します。コネクターは、インバウンド・メッセージの内容を使用して、これらの属性の値をビジネス・オブジェクト内に設定します。

HTTP(S) プロトコルの場合、プロトコル構成 MO の属性は次のとおりです。

表 10. イベント処理用 HTTP/HTTPS プロトコル構成 MO 属性

属性	必須	型	説明
Content-Type	なし	String	この属性の値は、発信メッセージの Content-Type ヘッダーを定義します (メッセージの ContentType および発信メッセージの 0 個以上の文字セットのパラメーターが含まれます)。構文は、HTTP プロトコルの Content-Type ヘッダーのものと同じです。例 : `text/xml; charset=ISO-8859-4`。Content-Type 属性が定義されていない場合、コネクターは、応答/障害メッセージの ContentType として要求の ContentType を使用します。
UserDefinedProperties	なし	ビジネス・オブジェクト	この属性は、ユーザー定義プロトコル・プロパティーのビジネス・オブジェクトを保持します。
1 つ以上の HTTP ヘッダー	なし	String	この属性を使用すると、ハンドラーは、指定された HTTP ヘッダーの値を受け渡したり、検索したりできます。
Authorization_UserId	なし	`String`	この属性は HTTP 基本認証の userID に対応します。
Authorization_Password	なし	`String`	この属性は HTTP 基本認証のパスワードに対応します。

これらの属性は、以下で説明されています。

- 『イベント処理のユーザー定義プロパティー』
- 27 ページの『イベント処理用の HTTP 証明書伝搬』

プロトコル・リスナーの詳細については、48 ページの『プロトコル・リスナー』を参照してください (要求処理用のプロトコル構成 MO については、31 ページの『同期要求処理 TLO』を参照してください)。

イベント処理のユーザー定義プロパティー: オプションとして、HTTP(S) プロトコル構成 MO でカスタム・プロパティーを指定できます。これを行うには、UserDefinedProperties 属性を組み込みます。この属性は、プロパティー値を持つ 1 つ以上の子属性を持つビジネス・オブジェクトに対応します。このビジネス・オブジェクトの各属性は、以下のようにメッセージ・ヘッダーの可変部分で読み取り (または、同期応答には書き込み) を行う単一プロパティーを定義する必要があります。

- 属性のタイプは、プロトコルのプロパティー・タイプに無関係に必ず String でなければなりません。属性のアプリケーション固有情報には、属性をマップするプロトコルのメッセージ・プロパティーの名前と形式を定義する 2 つの名前と値のペアを含めることができます。

表 11 は、これらの属性のアプリケーション固有情報を要約したものです。

表 11. ユーザー定義プロトコル・プロパティー属性のアプリケーション固有情報: 名前=値 ペアの内容

名前	値	説明
ws_prop_name (大文字小文字を区別しない。指定されない場合、プロパティー名として属性名が使用されます)	任意の有効なプロトコル・プロパティー名	これはプロトコル・プロパティーの名前です。ベンダーによっては、拡張機能を提供するために特定のプロパティーを予約している場合があります。

指定されたカスタム・プロパティー ASI (ws_prop_name) が無効で、このヘッダーを処理する論理的方法がない場合、コネクターは警告をログに記録し、このプロパティーを無視します。ws_prop_name に対する必要な検査が行われた後に、カスタム・プロパティーの値が設定も検索もされない場合、コネクターはエラーをログに記録し、イベントに失敗します。

UserDefinedProperties 属性が指定される場合、コネクターは、UserDefinedProperties ビジネス・オブジェクトのインスタンスを作成します。次に、コネクターは、メッセージからプロパティー値を抽出し、ビジネス・オブジェクトに保管しようとします。少なくとも 1 つのプロパティー値が正常に検索されると、コネクターは、変更された UserDefinedProperties 属性をプロトコル構成 MO に設定します。

同期イベント処理の場合、UserDefinedProperties 属性が指定され、そのビジネス・オブジェクトがインスタンス化されると、コネクターは、この子ビジネス・オブジェクトの各属性を処理し、それに応じてメッセージ・プロパティー値を設定します。

イベント処理用の HTTP 証明書伝搬: 証明書伝搬のために、コネクターは HTTP プロトコル構成 MO の Authorization_UserId 属性および Authorization_Password 属性をサポートします。このサポートは、HTTP 基本認証スキームの一部としてのこれらの証明書の伝搬に限られます。

HTTP または HTTPS プロトコル・リスナーが許可ヘッダーを含む HTTP サービス要求を処理する場合は、リスナーはヘッダーを解析して HTTP 基本認証に従うかどうか判別します。従う場合は、リスナーはユーザー名およびパスワードを抽出し、Base64 を使用してデコードします。このデコードされたストリングは、コロンで区切られたユーザー名およびパスワードから構成されます。プロトコル・リスナーがプロトコル構成 MO で Authorization_UserId 属性および Authorization_Password 属性を検出した場合、リスナーは、イベント許可ヘッダーから抽出した値をこれらの属性に設定します。

非同期イベント処理 TLO

図 7 は、非同期イベント処理のためのビジネス・オブジェクト階層を示しています。要求オブジェクトのみが必須です。

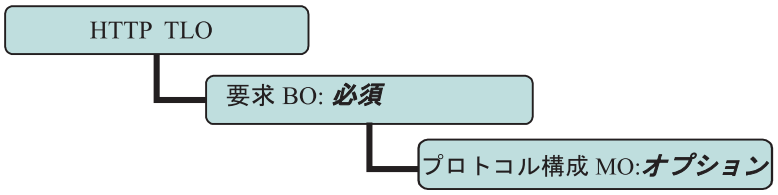


図 7. 非同期イベント処理のためのビジネス・オブジェクト階層

TLO には、オブジェクト・レベルの ASI のほか、属性レベルの ASI を持った属性が含まれています。両方の種類の ASI について、以下で説明します。

非同期イベント処理 TLO のためのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI は、TLO の性質、および TLO に含まれるオブジェクトについての基本的情報を提供します。図 8 は、非同期イベント処理のためのサンプル TLO である SERVICE_ASYNCH_TLO_Order のオブジェクト・レベルの ASI を表しています。

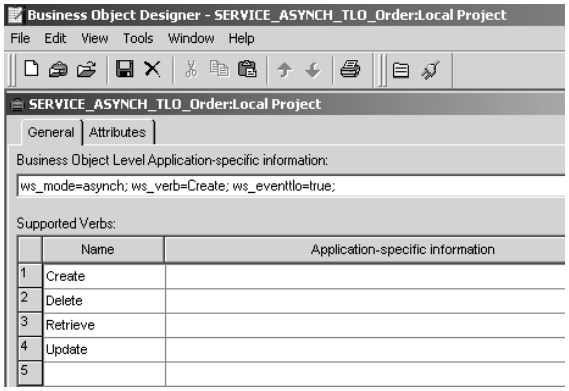


図 8. 非同期イベント処理のためのトップレベル・ビジネス・オブジェクト

以下の表 5 では、非同期イベント処理 TLO のためのオブジェクト・レベルの ASI について説明します。

表 12. 非同期イベント処理 TLO のオブジェクト ASI

オブジェクト・レベルの ASI	説明
ws_eventtlo=true	この ASI プロパティーが true に設定されている場合には、コネクターはこのオブジェクトをイベント処理用の TLO として扱います。
ws_verb=*verb*	コネクターは、TLO をコラボレーションに引き渡す前に、TLO で動詞を設定するためにこの ASI を使用します。図 8 に示されているサンプルでは、動詞は Create です。
ws_mode=asynch	イベント通知の際に、コネクターは、この ASI プロパティーを使用して、コラボレーションを同期 (synch) で呼び出すのか非同期 (asynch) で呼び出すのかを決定します。非同期処理の場合は、この ASI を asynch に設定する必要があります。 デフォルトは asynch です。

注: 同期イベント処理の場合とは異なり、非同期イベント処理では、TLO レベルのコラボレーション名 ASI は必要ありません。代わりに、統合ブローカーが、アプリケーション・イベントがそのような BO と動詞の組み合わせにサブスクライブするすべてのコラボレーションに到達できるようにします。

非同期イベント処理 TLO のための属性レベルの ASI

それぞれの非同期イベント処理 TLO には、要求ビジネス・オブジェクトに対応する単一の属性が含まれています。図 9 は、サンプル TLO の SERVICE_ASYNCH_TLO_Order の要求属性、および属性の ASI を表しています。

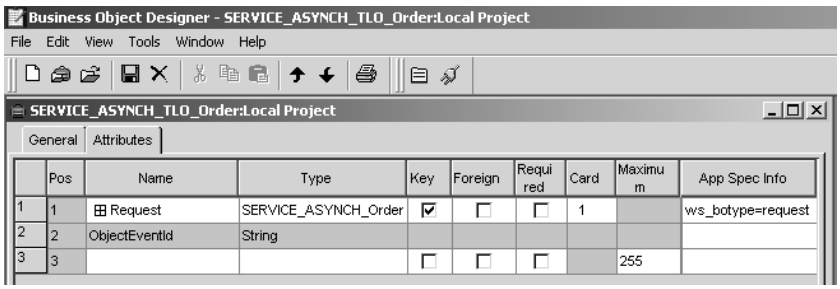


図 9. 非同期イベント処理のための TLO 属性

表 13 は、非同期イベント処理 TLO の要求属性に対する属性レベルの ASI を要約したものです。

表 13. 非同期イベント処理 TLO の属性 ASI

TLO 属性	属性レベルの ASI	説明
Request	ws_botype=request	この属性は、要求に対応します。コネクターはこの ASI を使用して、この TLO 属性のタイプが要求 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。複数の要求属性がある場合は、コネクターは最初の要求属性の ASI を使用します。 この属性は、同期イベント処理 TLO の場合には必須です。

非同期イベント処理のための要求ビジネス・オブジェクト

要求ビジネス・オブジェクトは、TLO の子であり、非同期イベント処理の場合には必須です。要求ビジネス・オブジェクトに対して、デフォルトの動詞を指定できます。これを行うには、トップレベルの要求ビジネス・オブジェクトの「サポートされている動詞 (Supported Verbs)」リストにある動詞の ASI フィールドに、

```
DefaultVerb=true;
```

と指定します。DefaultVerb ASI が指定されず、データ・ハンドラーが動詞の設定されていないビジネス・オブジェクトを処理する場合、そのビジネス・オブジェクトは、動詞なしで戻されます。非同期イベント処理のための要求ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 14 に説明されています。

表 14. 非同期イベント処理: 要求ビジネス・オブジェクトのためのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
cw_mo_http=*HTTPCfgMO*	この ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。この ASI は、HTTP または HTTPS プロトコル・リスナーを指定します。ASI およびプロトコル構成 MO は、両方ともオプションです。詳しくは、25 ページの『プロトコル構成 MO』を参照してください。注: ビジネス・オブジェクト変換用に構成するデータ・ハンドラーは、cw_mo で始まる ASI を、変換するビジネス・データの一部としてではなく、メタデータとして読み取ることができる必要があります。XML データ・ハンドラーには、cw_mo メタデータを検出し、その値が指示する属性を無視する機能があります。

表 14. 非同期イベント処理: 要求ビジネス・オブジェクトのためのオブジェクト・レベルの ASI (続き)

オブジェクト・レベルの ASI	説明
ws_tloname=*tloname*	この ASI は、このオブジェクトが属する TLO の名前を指定します。イベント処理の際に、コネクターは、この ASI を使用して、データ・ハンドラーによって引き渡された要求ビジネス・オブジェクトが TLO の子であるかどうかを判別します。引き渡された要求ビジネス・オブジェクトが TLO の子である場合は、コネクターは、指定された TLO を作成し、要求ビジネス・オブジェクトをその TLO の子として設定し、TLO のオブジェクト・レベルの ASI を使用して、この要求ビジネス・オブジェクトを、サブスクライブしているコラボレーションに引き渡します。

同期要求処理 TLO

要求処理のために、コネクターでは、同期および非同期の 2 種類の TLO を使用することができます。このセクションでは、同期要求処理 TLO について説明します。

図 10 は、同期要求処理のための TLO ビジネス・オブジェクト階層を示しています。要求オブジェクト、応答オブジェクト、およびハンドラー・オブジェクトは必須であり、障害オブジェクトはオプションです。イベント処理とは異なり、プロトコル構成 MO は、要求オブジェクトの場合には必須であり、応答および障害オブジェクトの場合には、オプションです。

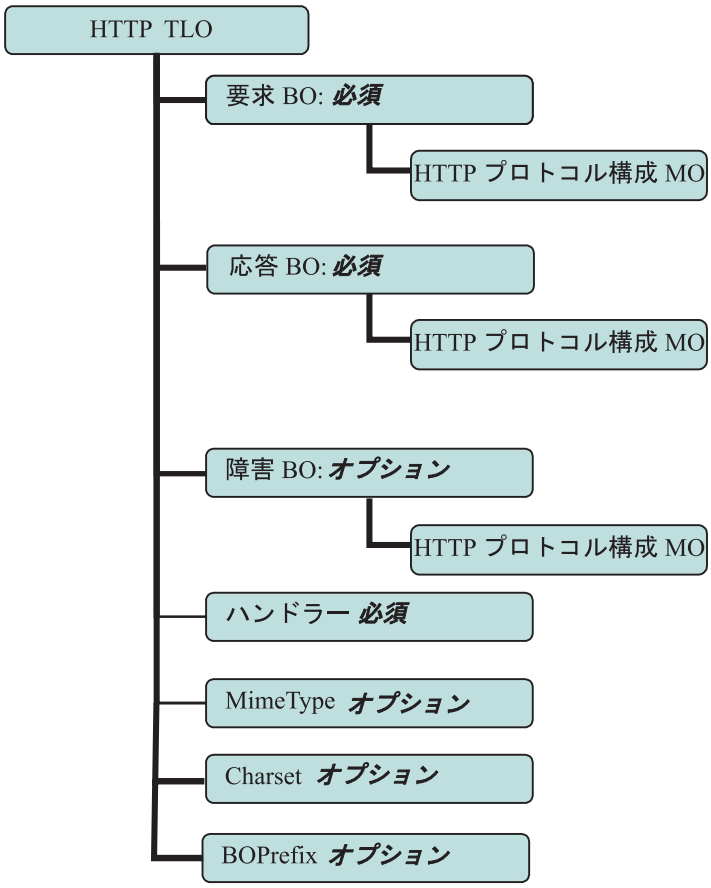


図 10. 同期要求処理のためのビジネス・オブジェクト階層

同期要求処理 TLO のオブジェクト・レベルの ASI

オブジェクト・レベルの ASI は、TLO の性質、および TLO に含まれるオブジェクトについての重要情報を提供します。図 11 は、同期要求処理のためのサンプル TLO である CLIENT_SYNCH_TLO_OrderStatus を表しています。

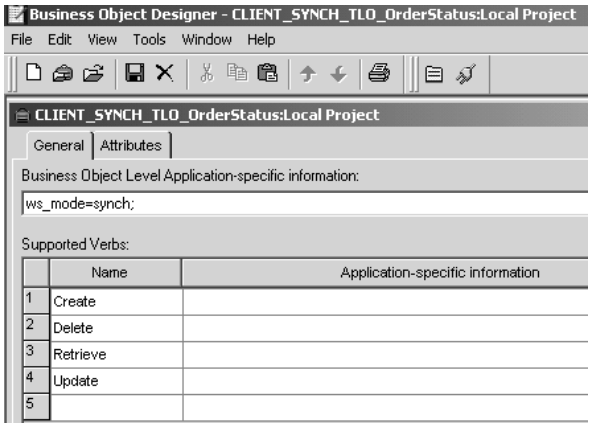


図 11. 同期要求処理のためのトップレベル・ビジネス・オブジェクト

表 15 では、同期要求処理 TLO のためのオブジェクト・レベルの ASI について説明します。同期イベント処理 TLO のための ASI とは異なり、ws_collab、ws_verb、または ws_eventtlo ASI は、このレベルでの要求処理には必須ではありません。

表 15. 同期要求処理 TLO のオブジェクト ASI

オブジェクト・レベルの ASI	説明
ws_mode=synch	要求処理の際に、コネクターは、この ASI プロパティーを使用して、HTTP サービスを同期 (synch) で呼び出すのか非同期 (asynch) で呼び出すのかを決定します。synch が指定された場合、コネクターは応答を予期します。したがって、TLO には、要求および応答ビジネス・オブジェクト、およびオプションとして 1 つまたは複数の障害オブジェクトを組み込む必要があります。 デフォルトは asynch です。

同期要求処理 TLO のための属性レベルの ASI

表 16 は、同期要求処理 TLO の属性および ASI について説明しています。

表 16. 要求処理の TLO 属性

TLO 属性	属性レベルの ASI	説明
MimeType	なし	この属性は、要求ビジネス・オブジェクトを要求メッセージに変換するためにコネクターが呼び出すデータ・ハンドラーの MIME タイプを指定します。メッセージ変換ルールの構成によっては、この値は、同期応答/障害メッセージをビジネス・オブジェクトに変換するときに使用される場合があります。
BOPrefix	なし	String 型のこの属性は、データ・ハンドラーに渡されます。
Handler	なし	この属性は、要求の処理に使用するプロトコル・ハンドラーを指定するもので、要求処理専用です。HTTP-HTTPS プロトコル・ハンドラーを指定する値 http を取ります。デフォルトは http です。
Charset		String 型のこのオプショナル・パラメーターは、要求ビジネス・オブジェクトをメッセージに変換するときに、データ・ハンドラーに設定される文字セットを指定します。注: この属性に指定される文字セット値は、要求メッセージの Content-Type プロトコル・ヘッダーには伝搬しません。

表 16. 要求処理の TLO 属性 (続き)

TLO 属性	属性レベルの ASI	説明
Request	`ws_botype=request`	この属性は、要求ビジネス・オブジェクトに対応します。コネクターはこの属性 ASI を使用して、この TLO 属性のタイプが要求 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。複数の要求属性がある場合は、コネクターは最初に取り込まれた属性の ASI を使用します。
Response	`ws_botype=response`	この属性は、コラボレーションによって戻される応答に対応し、同期要求処理の場合には必須です。コネクターはこの ASI を使用して、この TLO 属性のタイプが応答 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。

Fault	`ws_botype=fault`
	または
	`ws_botype=defaultfault`
		この属性は同期要求処理の場合のオプションであり、応答を正常に取り込むことができないときに HTTP サービスから戻される障害メッセージに対応するものです。
		
		コネクターはこの ASI を使用して、TLO の属性のタイプが障害 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。障害メッセージが詳細要素である場合は、`defaultfault` ビジネス・オブジェクトが戻されます。`defaultfault` は、デフォルトのビジネス・オブジェクトの解決に使用されます。

同期要求処理のための要求ビジネス・オブジェクト

要求ビジネス・オブジェクトは、TLO の子であり、同期要求処理の場合には必須です。要求ビジネス・オブジェクトは、オブジェクト・レベルの ASI を保有しています。

35 ページの表 17 では、同期要求処理のための要求ビジネス・オブジェクトのオブジェクト・レベルの ASI について説明します。

表 17. 同期要求処理: 要求ビジネス・オブジェクトのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
cw_mo_http=*HTTPCfgMO*	このオプショナル ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。このプロトコル構成 MO は、HTTP-HTTPS プロトコル・ハンドラーの宛先を指定します。この ASI は、HTTP-HTTPS プロトコル・ハンドラーによって使用されます。 TLO 要求属性は、要求処理用の HTTP プロトコル構成 MO を持たなければならないことに注意してください。詳しくは、 36 ページの『要求処理のための HTTP プロトコル構成 MO』を参照してください。注: ビジネス・オブジェクト変換用に構成するデータ・ハンドラーは、cw_mo で始まる ASI を、変換するビジネス・データの一部としてではなく、メタデータとして読み取ることができる必要があります。XML データ・ハンドラーには、cw_mo メタデータを検出し、その値が指示する属性を無視する機能があります。

同期要求処理のための応答ビジネス・オブジェクト

応答ビジネス・オブジェクトは、TLO の子であり、同期要求処理の場合には必須です。同期要求処理のための応答ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 18 に説明されています。

表 18. 同期要求処理: 応答ビジネス・オブジェクトのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
cw_mo_http=*HTTPCfgMO*	この ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。応答ビジネス・オブジェクトでオプションであるこのプロトコル構成 MO は、HTTP(s) プロトコル・ハンドラー用の応答メッセージのヘッダーを指定します。詳しくは、 36 ページの『要求処理のための HTTP プロトコル構成 MO』を参照してください。注: ビジネス・オブジェクト変換用に構成するデータ・ハンドラーは、cw_mo で始まる ASI を、変換するビジネス・データの一部としてではなく、メタデータとして読み取ることができる必要があります。XML データ・ハンドラーには、cw_mo メタデータを検出し、その値が指示する属性を無視する機能があります。

応答ビジネス・オブジェクトに対して、デフォルトの動詞を指定できます。これを行うには、トップレベルの要求ビジネス・オブジェクトの「サポートされている動詞 (Supported Verbs)」リストにある動詞の ASI フィールドに、

```
DefaultVerb=true;
```

と指定します。DefaultVerb ASI が指定されず、データ・ハンドラーが動詞の設定されていないビジネス・オブジェクトを処理する場合、応答ビジネス・オブジェクトが、動詞なしで戻されます。

同期要求処理のための障害ビジネス・オブジェクト

障害ビジネス・オブジェクトは、TLO の子であり、同期要求処理の場合にはオプションです。同期要求処理のための障害ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 9 に説明されています。

表 19. 同期要求処理: 障害ビジネス・オブジェクトのためのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
cw_mo_http=*HTTPCfgMO*	この ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。障害ビジネス・オブジェクトでオプションであるこのプロトコル構成 MO は、HTTP-HTTPS プロトコル・ハンドラー用の応答メッセージのヘッダーを指定します。詳しくは、 25 ページの『プロトコル構成 MO』を参照してください。注: ビジネス・オブジェクト変換用に構成するデータ・ハンドラーは、cw_mo で始まる ASI を、変換するビジネス・データの一部としてではなく、メタデータとして読み取ることができる必要があります。XML データ・ハンドラーには、cw_mo メタデータを検出し、その値が指示する属性を無視する機能があります。

要求処理のための HTTP プロトコル構成 MO

要求処理の際に、HTTP-HTTPS プロトコル・ハンドラーは、ターゲット HTTP サービスの宛先を判別するために、HTTP プロトコル構成 MO を使用します。このプロトコル構成 MO は、要求ビジネス・オブジェクトの場合には必須です。
HTTP-HTTPS プロトコル・ハンドラーは、HTTP 1.0 POST 要求のみをサポートします。 表 20 に示すように、唯一の必須属性 (Destination) はターゲット HTTP サービスの完全 URL です。オプションの許可属性については、以下のセクションで説明します。

表 20. 要求処理のための HTTP プロトコル構成 MO 属性

属性	必須	型	説明
Destination	はい	String	ターゲット HTTP サービスの宛先 URL。HTTP-HTTPS プロトコル・ハンドラーは、この属性を使用して、HTTP サービスの宛先を判別します。
Content-Type	要求ビジネス・オブジェクトでは必須。それ以外はオプション。	String	この属性の値は、発信メッセージの Content-Type ヘッダーを定義します (メッセージの ContentType およびオプションで発信メッセージの文字セットが含まれます)。構文は、HTTP プロトコルの Content-Type ヘッダーのものと同じです。例 : text/xml; charset=ISO-8859-4。

表 20. 要求処理のための HTTP プロトコル構成 MO 属性 (続き)

属性	必須	型	説明
Authorization_UserId	なし	String	この属性は HTTP 基本認証の userID に対応します。詳細については、 39 ページの『要求処理用の HTTP 証明書伝搬』を参照してください。
Authorization_Password	なし	String	この属性は HTTP 基本認証のパスワードに対応します。詳細については、 39 ページの『要求処理用の HTTP 証明書伝搬』を参照してください。
1 つ以上の HTTP ヘッダー	なし	String	この属性を使用すると、ハンドラーは、指定された HTTP ヘッダーの値を受け渡したり、検索したりできます。
UserDefinedProperties	なし	ビジネス・オブジェクト	この属性は、ユーザー定義プロトコル・プロパティーのビジネス・オブジェクトを保持します。詳細については、『要求処理のユーザー定義プロパティー』を参照してください。
MessageTransformationMap	なし	単一カーディナリティー・ビジネス・オブジェクト	0 個以上のメッセージ変換ルールを保持するビジネス・オブジェクトを指示する属性です。このルールは、着信メッセージに適用される、ルールで指定された MIME タイプおよび文字セットに関する情報を保持しています。詳細については、 38 ページの『メッセージ変換マップ』を参照してください。

HTTP プロトコル構成 MO 属性は、以下で説明されています。

- 『要求処理のユーザー定義プロパティー』
- 38 ページの『メッセージ変換マップ』
- 39 ページの『要求処理用の HTTP 証明書伝搬』

要求処理のユーザー定義プロパティー: オプションとして、HTTP プロトコル構成 MO でカスタム・プロパティーを指定できます。これを行うには、UserDefinedProperties 属性を組み込みます。この属性は、プロパティー値を持つ 1 つ以上の子属性を持つビジネス・オブジェクトに対応します。このビジネス・オブジェクトの各属性は、以下のようにメッセージ・ヘッダーの可変部分で読み取り (または、同期応答には書き込み) を行う単一プロパティーを定義する必要があります。

- 属性のタイプは、常に String でなければなりません。 属性のアプリケーション固有情報には、属性をマップするプロトコルのメッセージ・プロパティーの名前を定義する名前と値のペアを含めることができます。

表 21 は、これらの属性のアプリケーション固有情報を要約したものです。

表 21. ユーザー定義プロトコル・プロパティー属性のアプリケーション固有情報: 名前=値 ペアの内容

名前	値	説明
ws_prop_name (大文字小文字を区別しない。指定されない場合、プロパティー名として属性名が使用されます)	任意の有効なプロトコル・プロパティー名	これはプロトコル・プロパティーの名前です。ベンダーによっては、拡張機能を提供するために特定のプロパティーを予約している場合があります。

指定されたカスタム・プロパティー ASI (ws_prop_name) が無効で、このヘッダーを処理する論理的方法がない場合、コネクターは警告をログに記録し、このプロパティーを無視します。ws_prop_name に対する必要な検査が行われた後に、カスタム・プロパティーの値が設定も検索もされない場合、コネクターはエラーをログに記録し、イベントに失敗します。

UserDefinedProperties 属性が指定され、そのビジネス・オブジェクトがインスタンス化されると、コネクターは、この子ビジネス・オブジェクトの各属性を処理し、それに応じてメッセージ・プロパティー値を設定します。

同期要求処理で UserDefinedProperties 属性が指定されている場合、コネクターは、応答メッセージを受け取ると、UserDefinedProperties ビジネス・オブジェクトのインスタンスを作成し、メッセージからプロパティー値を抽出してから、それらを新規ビジネス・オブジェクトに保管しようとします。少なくとも 1 つのプロパティー値が正常に検索されると、コネクターは、変更された UserDefinedProperties ビジネス・オブジェクトをプロトコル構成 MO に設定します。

メッセージ変換マップ: メッセージ変換マップ (MTM) 機能は、要求処理 HTTP(S) プロトコル・ハンドラーのみでサポートされています。MessageTransformationMap はプロトコル構成 MO のオプション属性であり、ビジネス・オブジェクトを指示します。ビジネス・オブジェクトには、ルールで指定された MIME タイプおよび文字セットを使用してメッセージを変換するルールが含まれています。属性名 MessageTransformationMap (大文字小文字の区別がある) を見つけ、この属性のタイプがビジネス・オブジェクトである場合、コネクターは、そのオブジェクトのルールを使用してメッセージを変換します。

MTM 属性は、TransformationRule という名前の、カーディナリティー N の 1 つの子ビジネス・オブジェクト属性を持つ必要があります。メッセージの TransformationRule を見つける場合、HTTP-HTTPS プロトコル・ハンドラーは、まず、すべての TransFormationRule で指定されている ContentType によって正確にメッセージを一致させようとします。失敗すると、コネクターが、複数のタイプのメッセージに適用されるルールを見つけようとします。プロトコル・ハンドラー処理の詳細については、57 ページの『HTTP-HTTPS プロトコル・ハンドラー処理』を参照してください。

TransformationRule ビジネス・オブジェクトの各インスタンスには、表 22 に示すように、属性を指定する必要があります。

表 22. *HTTP プロトコル構成 MO の MessageTransformationMap 用の TransformationRule 属性*

属性名	必須	型	デフォルト値	説明
TransformationRule	なし	ビジネス・オブジェクト、カーディナリティー N		メッセージ変換用ルールを 1 つ保持する属性です。MessageTranformationMap 属性には、この属性の 0 個以上のインスタンスが存在する可能性があります。
+ContentType	はい	String	*/*	このプロパティーの値は、この変換ルールが適用されるメッセージの HTTP ContentType を指定します。この属性のデフォルト値 */* を使用すると、コネクターは、このルールを任意の ContentType に適用できます。プロトコル・ハンドラー処理の詳細については、57 ページの『HTTP-HTTPS プロトコル・ハンドラー処理』を参照してください。プロトコル・ハンドラーは、他のルールと同じ ContentType を持った複数のルールを見つけると、警告をログに記録し、重複したすべてのルールを無視し、固有なルールを使用することに注意してください。
+MimeType	なし			このビジネス・オブジェクトで指定された ContentType のメッセージの処理中に、データ・ハンドラーの呼び出しで使用する MIME タイプ。
+Charset	なし			このビジネス・オブジェクトで指定された ContentType の要求を変換するときに使用する文字セット。

要求処理用の HTTP 証明書伝搬: 証明書伝搬のために、コネクターは HTTP プロトコル構成 MO の Authorization_UserId 属性および Authorization_Password 属性をサポートします。このサポートは、HTTP 基本認証スキームの一部としてのこれらの証明書の伝搬に限られます。

コラボレーションは、プロトコル構成 MO の Authorization_UserId 属性および Authorization_Password 属性の値を設定します。これらの属性が null でも空でもない場合は、コネクターは、ターゲット HTTP サービスに送信する要求に許可ヘッダーを作成します。HTTP/HTTPS プロトコル・ハンドラーは、許可ヘッダーの作成時に *HTTP Authentication: Basic and Digest Access Authentication (RFC 2617)* に従います。

注: ダイジェスト認証方式はサポートされていません。また、Rfc2617 で定義されている、HTTP 認証のオプションのチャレンジ応答機構もサポートされていませ

ん。 HTTP(s) プロトコル・ハンドラーが証明書を必要とするサーバーを呼び出す場合、コネクターは、サーバーからのチャレンジ応答を待ちません。その代わり、証明書を継続して送信します。

非同期要求処理 TLO

図 12 は、非同期要求処理のためのビジネス・オブジェクト階層を示しています。要求オブジェクトおよびハンドラー・オブジェクトが必須です。要求オブジェクトには、HTTP-HTTPS プロトコル・ハンドラーのプロトコル構成 MO が含まれています。これらについて、以下のセクションで説明します。

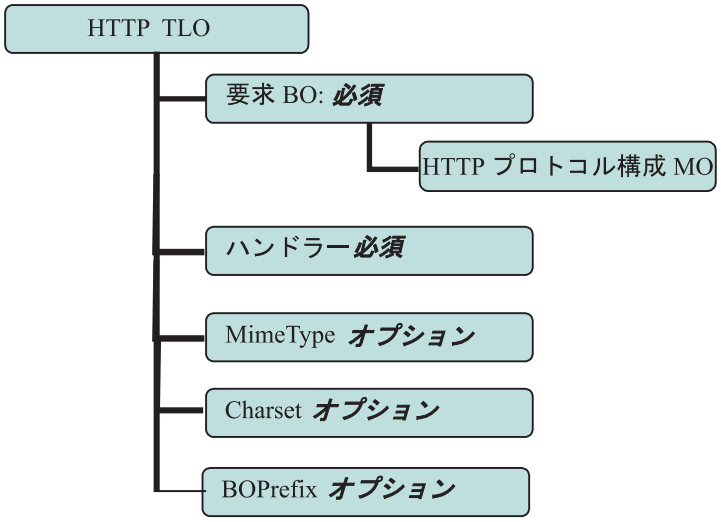


図 12. 非同期要求処理のためのビジネス・オブジェクト階層

TLO には、オブジェクト・レベルの ASI のほか、属性レベルの ASI を持った属性が含まれています。両方の種類の ASI について、以下で説明します。

非同期イベント処理 TLO のためのオブジェクト・レベルの ASI

図 13 は、非同期要求処理のためのサンプル TLO である CLIENT_ASYNCH_Order_TLO を表しています。

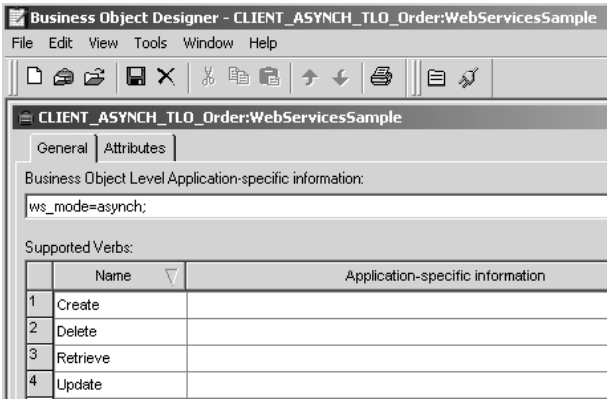


図 13. 非同期要求処理のためのトップレベル・ビジネス・オブジェクト

以下の表 23 では、非同期要求処理 TLO のオブジェクト・レベルの ASI について説明します。

表 23. 非同期要求処理 TLO のオブジェクト ASI

オブジェクト・レベルの ASI	説明
ws_mode=asynch	要求処理の際に、コネクターは、この ASI プロパティーを使用して、コラボレーションを同期 (synch) で呼び出すのか非同期 (asynch) で呼び出すのかを決定します。非同期要求処理の場合は、この ASI を asynch に設定する必要があります。 デフォルトは asynch です。

非同期要求処理 TLO のための属性レベルの ASI

表 24 は、非同期要求処理 TLO の要求属性に対する属性レベルの ASI を要約したものです。

表 24. 非同期要求処理 TLO の属性

TLO 属性	属性レベルの ASI	説明
MimeType	なし	この属性は、コネクターが呼び出すデータ・ハンドラーの MIME タイプを指定します。この属性は要求処理のみに使用されます。
BOPrefix	なし	この属性の値はデータ・ハンドラーに渡されます。
Handler	なし	この属性は、要求の処理に使用するプロトコル・ハンドラーを指定するもので、要求処理専用です。要求を処理する HTTP-HTTPS プロトコル・ハンドラーを指定する値 http を取ります。デフォルトは http です。
Charset		String 型のこのオプショナル・パラメーターは、要求ビジネス・オブジェクトをメッセージに変換するときに、データ・ハンドラーに設定される文字セットを指定します。注: この属性に指定される文字セット値は、要求メッセージの Content-Type プロトコル・ヘッダーには伝搬しません。

表 24. 非同期要求処理 TLO の属性 (続き)

TLO 属性	属性レベルの ASI	説明
Request	ws_botype=request	この属性は、HTTP サービス要求ビジネス・オブジェクトに対応します。コネクターはこの属性 ASI を使用して、この TLO 属性のタイプが要求 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。複数の要求属性がある場合は、コネクターは最初の要求属性の ASI を使用します。

非同期要求処理のための要求ビジネス・オブジェクト

要求ビジネス・オブジェクトは、TLO の子であり、非同期要求処理の場合には必須です。非同期要求処理のための要求ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 25 に説明されています。

表 25. 非同期要求処理: 要求ビジネス・オブジェクトのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
cw_mo_http=*HTTPCfgMO*	この ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。このプロトコル構成 MO は、HTTP-HTTPS プロトコル・ハンドラーの宛先を指定します。この ASI は、HTTP-HTTPS プロトコル・ハンドラーによって使用されます。 TLO 要求属性は、要求処理用の HTTP プロトコル構成 MO を持たなければならないことに注意してください。詳しくは、 36 ページの『要求処理のための HTTP プロトコル構成 MO』を参照してください。注: ビジネス・オブジェクト変換用に構成するデータ・ハンドラーは、cw_mo で始まる ASI を、変換するビジネス・データの一部としてではなく、メタデータとして読み取ることができる必要があります。XML データ・ハンドラーには、cw_mo メタデータを検出し、その値が指示する属性を無視する機能があります。

非同期要求処理のためのプロトコル構成 MO

要求処理の際に、HTTP-HTTPS プロトコル・ハンドラーは、ターゲット HTTP サービスの宛先を判別するために、HTTP プロトコル構成 MO を使用します。このプロトコル構成 MO は、要求ビジネス・オブジェクトの場合には必須です。詳しくは、 36 ページの『要求処理のための HTTP プロトコル構成 MO』を参照してください。

ビジネス・オブジェクトの開発

ビジネス・オブジェクトを作成するには、Business Object Designer Express を使用し、ビジネス・オブジェクトをサポートするようにコネクターを構成するには、Connector Configurator Express を使用します。Business Object Designer Express ツールの詳細については、「*ビジネス・オブジェクト開発ガイド*」および 105 ページの『付録 B. Connector Configurator Express』を参照してください。

第 4 章 HTTP コネクター

この章では、HTTP コネクターとその構成方法について説明します。

すべての WebSphere Business Integration コネクターは、統合ブローカーと連動して動作します。HTTP コネクターは、InterChange Server Express 統合ブローカーと連動します。これについては、「システム・インプリメンテーション・ガイド」で説明されています。

コネクターは、アダプターのランタイム・コンポーネントです。コネクターは、アプリケーション固有のコンポーネントとコネクター・フレームワークからなります。アプリケーション固有のコンポーネントには、特定のアプリケーションに応じて調整されたコードが含まれます。コネクター・フレームワーク (このコードは、すべてのコネクターで共通です) は、統合ブローカーとアプリケーション固有のコンポーネントとの仲介役を果たします。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントとの間で以下のようなサービスを提供します。

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

本書には、アプリケーション固有のコンポーネントおよびコネクター・フレームワークに関する情報が記載されています。本書では、この 2 つのコンポーネントをまとめてコネクターと呼びます。

統合ブローカーとコネクターの関係に関する詳細については、「システム管理ガイド」を参照してください。

コネクター処理

コネクターには、イベント処理を行うプロトコル・リスナー・フレームワークと要求処理を行うプロトコル・ハンドラー・フレームワークがあります。コネクター・フレームワークでは、この双方向の機能を利用して以下のことが可能になります。

- HTTP クライアントからの呼び出しの処理 (イベント処理)
- HTTP サービスを呼び出すコラボレーションによる要求の処理 (要求処理)

イベント処理の概要

コネクター・イベント処理 (またはイベント通知) は、HTTP クライアントからの要求を処理するために使用します。このイベント処理機能は、以下のコンポーネントを含む、プロトコル・リスナー・フレームワークを採用しています。これらのコンポーネントについては、この章の後半でさらに詳しく説明します。

- HTTP プロトコル・リスナー
- HTTPS プロトコル・リスナー

コネクターは、これらのコンポーネントを使用して、トランスポートでクライアントからコラボレーションへの呼び出しを listen します。

クライアントからの要求が到着すると、リスナーは、要求メッセージをビジネス・オブジェクトに変換し、コラボレーションを呼び出します。これが同期要求である場合、コネクターは同一タイプの応答ビジネス・オブジェクトを、要求ビジネス・オブジェクトとして受信します。リスナーは、応答ビジネス・オブジェクトを応答メッセージに変換します。リスナーは、次に応答メッセージをクライアントにトランスポートします。イベントの順序付けは、このコネクターの要件ではないので注意してください。つまり、コネクターはイベントをどのような順序でも配信できます。

HTTP コネクターは、構成されたデータ・ハンドラーを使用して、着信した要求メッセージをビジネス・オブジェクトに変換します。着信した要求メッセージをどのビジネス・オブジェクトで解決すべきか、データ・ハンドラーが判別できるように、コネクターは、サポートされているビジネス・オブジェクトに関するメタ情報をデータ・ハンドラーに提供します。コネクターはまず最初に、サポートされているビジネス・オブジェクトの中から変換の候補となるすべてのビジネス・オブジェクトのリストを作成します。このリストは、サポートされている TLO のみで構成されます。 サポートされている TLO ビジネス・オブジェクトは、オブジェクト・レベル ASI が `ws_eventtlo=true` のビジネス・オブジェクトです。

プロトコル・リスナーは、以下のようにして TLO のオブジェクト・レベル ASI を読み取ります。

- `ws_collab=` これは、どのコラボレーションを呼び出すのかを決定します。
- `ws_mode=` これは、コラボレーションを呼び出す方法 (同期 (`synch`) または非同期 (`asynch`)) を決定します。

コネクターはデータ・ハンドラーから戻された要求ビジネス・オブジェクトを検査します。コネクターは、このビジネス・オブジェクトの `ws_tloname` ASI を使用して、親 TLO の名前を抽出します。この TLO はインスタンス化され、要求ビジネス・オブジェクトが TLO に設定されます。最後に、構成されたこの TLO がコラボレーションの呼び出しに使用されます。

コラボレーションを同期実行する場合、コネクターはデータ・ハンドラーを使用して、クライアントに戻す応答または障害メッセージを作成します。この場合、コネクターは、単に、ビジネス・オブジェクト (TLO の子) をデータ・ハンドラーに渡します。データ・ハンドラーは、渡されたビジネス・オブジェクトに基づいてメッセージを戻します。

要求処理の概要

コネクターは、コラボレーションのために、HTTP(S) を介して HTTP サービスを呼び出すことができます。この要求処理機能は、プロトコル・ハンドラー・フレームワークによってサポートされています。プロトコル・ハンドラー・フレームワークは、HTTP-HTTPS プロトコル・ハンドラーで構成される構成可能ランタイム・モジュールです。後から、この章でこれについて詳しく説明します。

プロトコル・ハンドラー・フレームワークは、コラボレーションの要求ビジネス・オブジェクト (常に TLO に設定されている) を受信すると、プロトコル・ハンドラーをロードします。プロトコル・ハンドラーは、HTTP サービスの呼び出しおよび (オプションで) 応答の保護に必要なトランスポート・レベルの詳細を管理し、コラボレーション要求ビジネス・オブジェクトの要求メッセージへの変換、要求メッセージによるエンドポイント Web サービスの呼び出し、および応答メッセージのビジネス・オブジェクトへの変換ならびにそのオブジェクトのコラボレーションへのリターン (要求/応答 (同期) モードの場合)、という 3 つのメインタスクを実行します。

HTTP コネクターは常に TLO を使用してコラボレーションから呼び出されます。コネクターは、TLO から要求ビジネス・オブジェクトを判別して、このビジネス・オブジェクトによりデータ・ハンドラーを呼び出します。データ・ハンドラーは、コネクターにより送信された要求メッセージを HTTP サービスに戻します。

同期実行する場合、コネクターはデータ・ハンドラーを使用して、応答および障害メッセージをそれぞれ応答ビジネス・オブジェクトおよび障害ビジネス・オブジェクトに変換します。データ・ハンドラーが、これらの応答/障害をビジネス・オブジェクトに変換する際に、どのビジネス・オブジェクトにより解決すべきかを判別できるようにするために、コネクターは、特定のメタ情報をデータ・ハンドラーに提供します。具体的には、コネクターは、呼び出し側の TLO の子であるすべての応答および障害ビジネス・オブジェクトのリストを作成します。応答ビジネス・オブジェクトは 1 つだけ存在している必要があり、オプションで多数の障害ビジネス・オブジェクトが存在します。デフォルトの障害ビジネス・オブジェクトが 1 つだけ存在している場合もあります。デフォルトの障害ビジネス・オブジェクトの場合、コネクターは、データ・ハンドラーにデフォルトの障害ビジネス・オブジェクトの名前を通知するだけです。この変換で解決する障害ビジネス・オブジェクトが他にない場合、最後の手段として、データ・ハンドラーによってデフォルトの障害ビジネス・オブジェクトを解決する必要があります。

HTTP(S) サービス

HTTP サービスは HTTP トランスポート・プロトコルをサポートします。HTTP は、HTTP クライアントが接続を開き、要求メッセージを HTTP サーバーに送信するというクライアント/サーバー・モデルを実現します。クライアント要求メッセージは HTTP サービスを呼び出します。HTTP サーバーは、呼び出しを含むメッセージをディスパッチして、接続を閉じます。

コネクターの HTTP および HTTPS プロトコル・リスナーは、コラボレーションに対するクライアント要求を処理する際に HTTP クライアント/サーバーおよび要求/応答モデルを利用します。ただし、HTTP リスナーは、HTTP サーバー (プロキシー、仲介、またはそれ以外) として機能するようになっていません。HTTP リスナ

ーは、どちらかといえば、企業内およびファイアウォールの内側で使用するエンドポイントとして機能します。したがって、クライアント要求をリスナーに送るには、ファイアウォール内に別の Web サーバーまたはゲートウェイを配置する必要があります。詳しくは、1 ページの『第 1 章 アダプターの概要』を参照してください。

同期 HTTP(S) サービス

コネクター処理の観点からは、同期 HTTP サービスは、要求/応答の流れをたどるサービスです。 HTTP または HTTPS プロトコル・リスナーにより HTTP 要求メッセージが正常に処理された場合、メッセージ本文には応答および HTTP 状況コード `200 OK` が記載されます。障害が戻された場合、本文には障害メッセージおよび状況コード `500` が記載されます。

非同期 HTTP(S) サービス

コネクター処理の観点からは、非同期 HTTP サービスは、要求専用の流れをたどるサービスです。HTTP または HTTPS プロトコル・リスナーにより、要求専用操作が正常に受信および処理された場合、HTTP 状況コード `202 Accepted` が生成されます。HTTP 状況コード `200 OK` が生成されるようにコネクターを構成することもできます。詳細については、表 33 の HTTPAsyncResponseCode プロパティーを参照してください。障害が発生すると、HTTP 状況コード `500` が生成されます。応答はありませんが、障害の本文が戻されることがあります。

イベント処理

コネクターは、イベント処理のときに、プロトコル・リスナーおよび構成済みデータ・ハンドラーを使用して、HTTP サービス・クライアントからの要求メッセージを、コラボレーションが取り扱うことのできるビジネス・オブジェクトに変換します。プロトコル・リスナーは、イベント処理において極めて重要な役割を果たしています。

プロトコル・リスナー

HTTP 要求は、HTTP または HTTPS のトランスポートによって届きます。リスナーは、トランスポート・チャネルにこのような要求が到着するのをモニターします。プロトコル・リスナーには次の 2 種類があり、それぞれに対応するチャネルがあります。

- HTTP プロトコル・リスナー
- HTTPS プロトコル・リスナー

これらの各リスナーは、トランスポートで listen するスレッドで構成されます。クライアントから要求メッセージを受け取ると、リスナーは、プロトコル・リスナー・フレームワークにそのイベントを登録します。

プロトコル・リスナー・フレームワークは、プロトコル・リスナーを管理し、リソースが使用可能になったときに要求を処理するようにスケジューリングします。コネクター固有のプロパティーに値を設定するときに、プロトコル・リスナー・フレームワークのリスナーおよび性質を構成してください。構成可能なプロトコル・リスナー・フレームワークのプロパティーには、以下のものが含まれます。

- **WorkerThreadCount** プロトコル・リスナー・フレームワークが使用することのできるスレッドの合計数。これは、プロトコル・リスナー・フレームワークが同時に処理できる要求の数です。
- **RequestPoolSize** プロトコル・リスナー・フレームワークに登録できる要求の最大数。この最大数を超える要求を受け取ると、新規要求は登録されなくなります。

これら 2 種類のコネクター固有のプロパティーは、プロトコル・リスナーが際限なくイベントを発生させてコネクターをふさいでしまわないように、メモリー割り振りを制御します。この割り振りアルゴリズムは、「コネクターは、WorkerThreadCount + RequestPoolSize に等しいイベントの合計数を常に受信できる」というものです。ここでは、*WorkerThreadCount* 数の要求を並行して処理できます。

追加のプロトコル・リスナーをプロトコル・リスナー・フレームワークに接続することができます。詳しくは、74 ページの『複数のプロトコル・リスナーの作成』および 64 ページの『コネクター固有の構成プロパティー』を参照してください。

HTTP および HTTPS プロトコル・リスナー処理

HTTP(S) プロトコル・リスナーは、クライアントの HTTP(S) 要求について継続的に listen するスレッドから構成されています。リスナー・スレッドは、コネクター固有の構成 (リスナー) プロパティー Host および Port で指定されているホストとポートをバインドします。別の構成プロパティー (RequestWaitTimeout) は、コネクターがシャットダウンされたかどうか確認するまでリスナーが要求を待つインターバルを定義します。

図 14 は、同期操作のための HTTP プロトコル・リスナー処理を表しています。

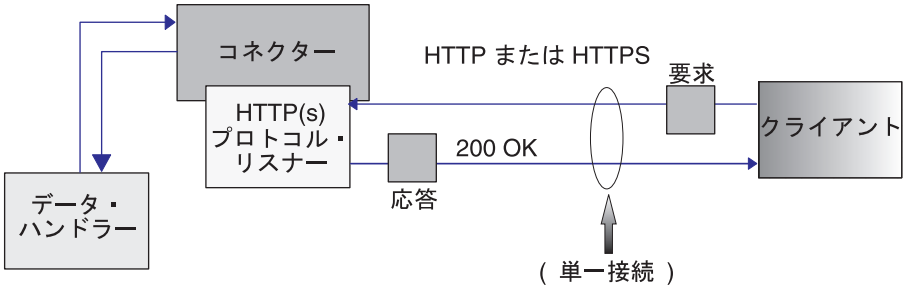


図 14. HTTP プロトコル・リスナー: 同期イベント処理

図 15 は、非同期操作のための HTTP プロトコル・リスナー処理を表しています。

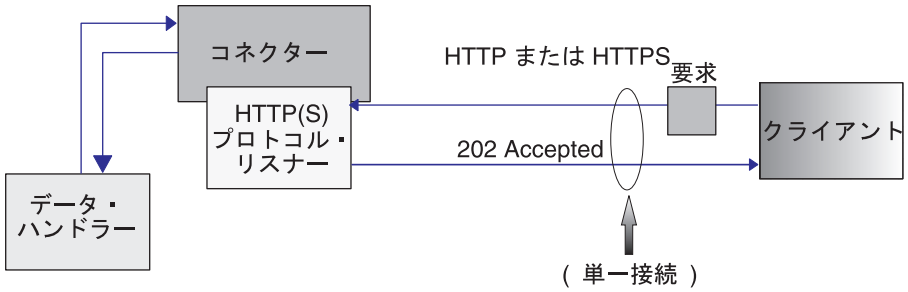


図 15. HTTP プロトコル・リスナー: 非同期イベント処理

クライアントは、HTTP または HTTPS 要求を開始する場合、要求メッセージを HTTP または HTTPS リスナーに通知します。プロトコル・リスナー URL を呼び出すためには、クライアントは HTTP POST メソッドを使用する必要があります。

HTTP(S) 要求が到着すると、リスナーは、要求をプロトコル・リスナー・フレームワークに登録し、リソースが使用可能になったときにイベントを処理するようにスケジュールします。リスナーは、次に、要求からプロトコル・ヘッダーとペイロードを抽出します。

表 26 は、リスナーが、インバウンド・メッセージの Charset、MimeType、ContentType、Content-Type ヘッダーを判別するときに使用するルールの優先順位の要約です。

表 26. HTTP(s) プロトコル・リスナーによる、インバウンド・メッセージの処理ルール

優先順位の順序	Charset	MimeType	ContentType	Content-Type ヘッダー
1	着信 HTTP メッセージの Content-Type ヘッダー値の Charset パラメーター値	このリスナーの URLsConfiguration コネクター・プロパティー値	Content-Type ヘッダー値の、着信 HTTP メッセージのタイプ/サブタイプの値	着信 HTTP メッセージの Content-Type ヘッダー
2	このリスナーの URLsConfiguration プロパティー値			
3	要求メッセージのタイプ ContentType がサブタイプの付いた `text` (`text/xml`、`text/plain` など) の場合、ISO-8859-1 にデフォルト設定されます。それ以外の場合、Charset は使用されません。	ContentType にデフォルト設定されます		

表 26 は次のことを示しています。

- プロトコル・リスナーは、以下のルールに従って、インバウンド・メッセージの Charset を判別します。

1. リスナーは、HTTP メッセージの Content-Type ヘッダー値の文字セット・パラメーターから Charset を抽出しようとします。
2. Content-Type ヘッダーから Charset 値を取得できない場合、プロトコル・リスナーは、このリスナーの URLsConfiguration プロパティー値を読み取ろうとします。
3. 前のステップで説明した方法で Charset 値が取得できない場合で、メッセージ・タイプ ContentType が、サブタイプの付いた text (text/xml、text/plain など) のときは、リスナーは、ISO-8859-1 のデフォルトの Charset 値を使用します。それ以外の場合、Charset 値は使用されません。

- リスナーは、以下のルールに従って、応答メッセージの MimeType を判別します。
  1. 着信要求メッセージによって使用される URL の TransformationRule が構成されており、要求 ContentType が TransformationRule の ContentType と一致する場合、リスナーは TransformationRule を使用して、要求メッセージを要求ビジネス・オブジェクトに変換するための MimeType を抽出します。リスナーは、要求された URL の URLsConfiguration プロパティーの ContentType (text/xml など) に基づいて、正確な TransformationRule 一致を見つけようとします。
  2. これに失敗すると、リスナーは、要求 URL で複数の ContentType (*/* など) に適用される TransformationRule を見つけようとします。
  3. ここまでのすべてのステップで MimeType を判別できない場合、データ・ハンドラーを呼び出し、要求メッセージを要求ビジネス・オブジェクトに変換するために、ContentType の値が MimeType として使用されます。
- リスナーは、着信 HTTP メッセージの Content-Type ヘッダーからタイプ/サブタイプを抽出して、ContentType を判別します。
- リスナーは、着信 HTTP メッセージの Content-Type ヘッダーから、Content-Type ヘッダーを判別します。

コラボレーションを非同期に呼び出す場合、リスナーは、要求ビジネス・オブジェクトを統合ブローカーに配信し、HTTP 状況コード 202 Accepted でクライアントに応答します。これでリスナー処理が完結します。

同期呼び出しの場合は、リスナーは同期させてコラボレーションを呼び出します。コラボレーションは応答ビジネス・オブジェクトを使用して応答します。

表 27 は、リスナーが、応答メッセージの
Charset、MimeType、ContentType、Content-Type ヘッダーを判別するときに使用するルールの優先順位の要約です。

表 27. HTTP(s) プロトコル・リスナーによる、アウトバウンド同期応答メッセージの処理ルール

優先順位の順序	Charset	MimeType	ContentType	Content-Type ヘッダー
1	Protocol ConfigMO Content-Type ヘッダー	TLO の MimeType プロパティー	Protocol ConfigMO Content-Type ヘッダー	Protocol ConfigMO Content-Type ヘッダー
2	TLO の Charset プロパティー値	要求メッセージの MimeType。ただし、要求と応答の ContentType が一致する場合のみ。	要求メッセージの ContentType	ContentType および Charset を使用して、Content-Type ヘッダーを構成
3	要求メッセージの Charset。ただし、要求と応答の ContentType が一致する場合のみ。	ContentType 値を MimeType として使用		
4	ContentType が `text/*` の場合、ISO-8859-1 にデフォルト設定される。それ以外の場合、Charset は使用されません。			

表 27 は次のことを示しています。

- リスナーは、以下のルールに従って、応答メッセージの Charset を判別します。
  1. Charset が応答ビジネス・オブジェクトのプロトコル構成 MO に指定されている場合、その値が使用されます。
  2. Charset 値が応答ビジネス・オブジェクトのプロトコル構成 MO ヘッダーに指定されていない場合、リスナーは、Charset が TLO で指定されているかどうかを検査します。
  3. Charset が TLO で指定されておらず、応答が要求と同じ ContentType を持つ場合、応答には、要求の Charset が使用されます。
  4. これまでのステップで応答の Charset 値を判別できず、メッセージ・タイプ部分の ContentType が、なんらかのサブタイプが付いた `text` (`text/xml`、`text/plain` など) の場合、リスナーは ISO-8859-1 のデフォルトの Charset 値を使用します。それ以外の場合、Charset 値は使用されません。
- リスナーは、以下のルールに従って、応答メッセージの MimeType を判別します。
  1. TLO の MimeType 属性
  2. TLO の MimeType 属性がなく、要求と応答の ContentType が一致する場合、リスナーは、応答メッセージに要求の MimeType を使用します。
  3. それ以外の場合、リスナーは ContentType 値を MimeType として使用します。
- リスナーは、以下のルールに従って、応答メッセージの ContentType を判別します。

1. Content-Type ヘッダーが応答ビジネス・オブジェクトのプロトコル構成 MO で指定されている場合、Content-Type ヘッダーのタイプ/サブタイプ部分が ContentType として使用されます。
2. Content-Type ヘッダーが応答ビジネス・オブジェクトのプロトコル構成 MO で指定されていない場合、リスナーは、判別した ContentType および Charset (応答メッセージで Charset が判別した場合) を使用して、Content-Type ヘッダーを構成します。

リスナーは HTTP プロトコル構成 MO を処理します。HTTP プロトコル構成 MO で渡されるヘッダー値が、要求応答イベントのコンテキストにおいて正しくなるようにするのは、コラボレーションの責任です。リスナーは、以下のルールに従って、標準ヘッダーおよびカスタム・プロパティーのデータを取り込みます。

1. リスナーは、特殊な属性 (ObjectEventId など) を無視するために、HTTP プロトコル構成 MO の各項目を調べます。
2. 空でない各ヘッダーが発信メッセージに置かれ、追加処理 (Content-Type ヘッダーなど) が行われます。
3. 上記の方法では、リスナーは、メッセージに非標準のヘッダーを設定する場合がありますが、メッセージが論理的または文法的に正しいかどうかは検査しないことに注意してください。
4. HTTP プロトコル構成 MO の UserDefinedProperties 属性に 1 つ以上のカスタム・プロパティーがある場合、リスナーは、それらを Entity Headers Section (最後のヘッダー・セクション) に追加します。カスタム・プロパティーの詳細については、26 ページの『イベント処理のユーザー定義プロパティー』を参照してください。

注: HTTP プロトコル構成 MO で、Connection、Trailer、Transfer-Encoding、Content-Encoding、Content-Length、Content-MD5、Content-Range のいずれかのヘッダーを指定すると、誤った HTTP メッセージになる可能性が非常に高くなります。

リスナーは、次に、データ・ハンドラーを呼び出して、コラボレーションによって戻された応答ビジネス・オブジェクトを応答メッセージに変換します。

リスナーは、応答メッセージをクライアントに配信し、`200 OK` HTTP 状況コードを組み込みます。コラボレーションにより障害ビジネス・オブジェクトが戻された場合は、障害メッセージに変換されます。この障害メッセージは、`500 Internal Server Error` HTTP コードと共にクライアントに配信されます。

リスナーは、次に、接続を閉じ、イベントを処理したスレッドは使用可能になります。

HTTP プロトコル・リスナーのサポートされていない処理機能

HTTP プロトコル・リスナーは、以下の機能はサポートしていません。

- キャッシング: プロトコル・リスナーは、HTTP 仕様 (RFC2616) で定義されているキャッシング機能は実行しません。
- プロキシー: プロトコル・リスナーは、HTTP 仕様 (RFC2616) で定義されているプロキシー機能は実行しません。

- 持続接続: プロトコル・リスナーは、HTTP 仕様 (RFC2616) で定義されている持続接続はサポートしていません。その代わりに、プロトコル・リスナーは、各 HTTP 接続の範囲を単一クライアント要求とみなし、サービス要求が完了すると、接続を閉じます。プロトコル・リスナーは、その接続を、別のサービス呼び出しに再使用しようとはしません。
- リダイレクト: プロトコル・リスナーは、リダイレクトはサポートしていません。
- 大規模ファイル転送: プロトコル・リスナーは、大規模ファイル転送には使用できません。その代わりに、参照によって、大規模ファイルの引き渡しを行うことができます。
- 状態管理: プロトコル・リスナーは、RFC2965 に記載されている HTTP 状態管理機構はサポートしていません。
- Cookies: プロトコル・リスナーは、Cookies はサポートしていません。

セキュア・ソケットを使用した HTTPS リスナー処理

HTTPS プロトコル・リスナー処理は HTTP プロトコル・リスナー処理のセクションで説明されているとおりですが、HTTPS ではセキュア・ソケットを使用するという点が異なります。詳しくは、 61 ページの『SSL』を参照してください。

イベントの永続性と送達

イベントの永続性は、プロトコルによって決まります。

- **HTTP プロトコル・リスナー** 永続性がなく、送達は保証されません
- **HTTPS プロトコル・リスナー** 永続性がなく、送達は保証されません

イベントの順序付け

コネクターは、任意の順序でイベントを配送することができます。

イベントのトリガー

イベント・トリガーのメカニズムは、プロトコル・リスナーの構成方法によって異なります。

- **HTTP プロトコル・リスナー** HTTP 接続要求の場合、listen は ServerSocket を介して行われます。
- **HTTPS プロトコル・リスナー** HTTPS 接続要求の場合、listen は ServerSocket 層を介して行われます。

注: コネクターは、Create (作成)、Update (更新)、Retrieve (検索) または Delete (削除) の区別を行いません。これらのイベントは、すべて同じ方法で扱われます。

イベントの検出

イベントの検出は、それぞれのプロトコル・リスナーによって行われます。イベント検出のメカニズムは、トランスポート、および各リスナーごとのコネクター固有プロパティーの構成方法に、完全に依存しています。これらのプロパティーの詳細については、 64 ページの『コネクター固有の構成プロパティー』を参照してください。

イベント状況

イベント状況はプロトコル・リスナーによって管理され、トランスポート、およびリスナーの構成方法によって異なります。

- **HTTP プロトコル・リスナー** HTTP は本来、非永続的で同期的なものです。したがって、イベント状況は維持されません。
- **HTTPS プロトコル・リスナー** HTTP は本来、非永続的で同期的なものです。したがって、イベント状況は維持されません。

イベントの検索

イベントの検索はプロトコル・リスナーによって管理され、トランスポート、およびリスナーの構成方法によって異なります。

- **HTTP プロトコル・リスナー** イベントの検索は、ソケットから HTTP 要求を取り出すことによって行われます。
- **HTTPS プロトコル・リスナー** イベントの検索は、ソケットから HTTP 要求を取り出すことによって行われます。

イベントのアーカイブ

イベントのアーカイブはプロトコル・リスナーによって管理され、トランスポート、およびリスナーの構成方法によって異なります。

- **HTTP プロトコル・リスナー** トランスポートが非永続的であり、同期的であるため、アーカイブは行われません。
- **HTTPS プロトコル・リスナー** トランスポートが非永続的であり、同期的であるため、アーカイブは行われません。

イベントのリカバリー

イベントのリカバリーはプロトコル・リスナーによって管理され、トランスポート、およびリスナーの構成方法によって異なります。

- **HTTP プロトコル・リスナー** トランスポートが非永続的であるため、イベント・リカバリーは行われません。
- **HTTPS プロトコル・リスナー** トランスポートが非永続的であるため、イベント・リカバリーは行われません。

要求処理

コネクターの要求処理機能を使用して、コラボレーションから HTTP サービスを呼び出すことができます。コネクターおよびその要求処理コンポーネント (プロトコル・ハンドラー・フレームワークおよびプロトコル・ハンドラー) を構成する必要があります。

コネクターは、実行時にビジネス・オブジェクトの形でコラボレーションから要求を受け取ります。ビジネス・オブジェクト (要求、およびオプションとして、応答および障害のビジネス・オブジェクト) は、HTTP サービスを使用するように構成されたコラボレーションによって発行される TLO に含まれます。TLO およびその子ビジネス・オブジェクトには、処理モード (同期または非同期) を指定する属性および ASI、データ・ハンドラー MIME タイプ、使用するプロトコル・ハンドラー

の種類、およびターゲットのアドレスが入っています。プロトコル・ハンドラーは、この情報を使用して、データ・ハンドラーのインスタンスを呼び出し、要求ビジネス・オブジェクトを要求メッセージに変換し、ターゲット HTTP サービスを呼び出します。同期モードの場合、プロトコル・ハンドラーは、データ・ハンドラーを再度呼び出し、応答メッセージを応答ビジネス・オブジェクトに変換して、これをコラボレーションに返送します。

コネクターは、要求メッセージに対する応答として、リモート側の取引先から次のいずれかを受け取ることができます。

- データを含んでいる応答メッセージ
- 障害情報を含んでいる応答メッセージ

プロトコル・ハンドラーは、要求処理において重要な役割を果たしています。

プロトコル処理

コラボレーションは、HTTP または HTTPS のトランスポートによって、HTTP サービスを呼び出します。コネクターは、1 つのプロトコル・ハンドラー (HTTP および HTTPS のサービスを呼び出す HTTP-HTTPS プロトコル・ハンドラー) と対応するチャネルを持っています。

プロトコル・ハンドラー・フレームワークはプロトコル・ハンドラーを管理し、起動時にプロトコル・ハンドラーをロードします。コネクターが要求ビジネス・オブジェクトを受け取ると、要求スレッド (それぞれのコラボレーション要求は、独自のスレッドで送られてきます) は、プロトコル・ハンドラー・フレームワークを呼び出して、要求を処理します。

プロトコル・ハンドラー・フレームワークは、TLOs Handler 属性 ASI を読み取り、使用するプロトコル・ハンドラーを決定します。一連のルールを適用して (57 ページの『HTTP-HTTPS プロトコル・ハンドラー処理』を参照)、プロトコル・ハンドラーはデータ・ハンドラーを呼び出し、要求ビジネス・オブジェクトを要求メッセージに変換します。プロトコル・ハンドラーは、要求メッセージをトランスポート (HTTP(S)) メッセージにパッケージします。

次に、プロトコル・ハンドラーは、要求ビジネス・オブジェクトのプロトコル構成 MO の Destination 属性を読み取り、ターゲット・アドレスを判別します。プロトコル・ハンドラーは、次に、要求メッセージを使用してターゲット HTTP サービスを呼び出します。

プロトコル・ハンドラーは、`ws_mode TLO` ASI を読み取り、処理モードが同期または非同期のいずれであるかを判別します。この ASI が `asynch` に設定されていると、プロトコル・ハンドラー処理は完了します。このように設定されていない場合、プロトコル・ハンドラーは応答メッセージを待ちます。応答メッセージが到着すると、プロトコル・ハンドラーは、プロトコル・ヘッダーとペイロードを抽出します。次にデータ・ハンドラー (MimeType TLO 属性によって指示される) を呼び出して、メッセージを応答または障害ビジネス・オブジェクトに変換します。プロトコル・ハンドラーは、プロトコル構成 MO を再度使用して、プロトコル・ヘッダーをビジネス・オブジェクトに設定します。プロトコル・ハンドラーは、次に、応答または障害ビジネス・オブジェクトをコラボレーションに戻します。

コネクターの構成によっては、1 つまたは複数のプロトコル・ハンドラーがコネクターにプラグされている場合があります。コネクター固有のプロパティーを指定することにより、プロトコル・ハンドラーを構成することができます。

HTTP-HTTPS プロトコル・ハンドラー処理

HTTP-HTTPS プロトコル・ハンドラーは、このセクションに記載している点を除き、56 ページの『プロトコル処理』で説明しているように動作します。図 16 は、同期操作のための HTTP-HTTPS プロトコル・ハンドラーを表しています。

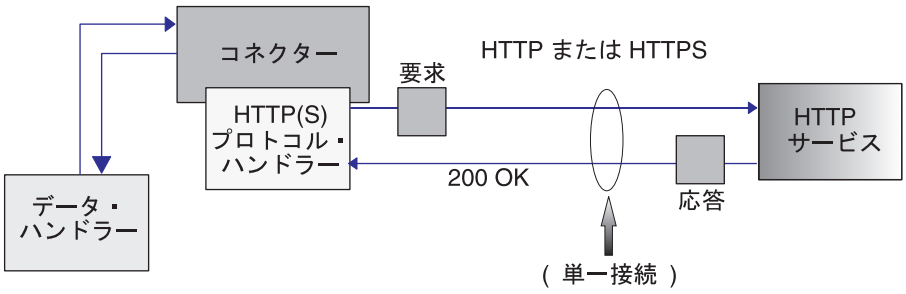


図 16. HTTP-HTTPS プロトコル・ハンドラー: 同期要求処理

図 17 は、非同期要求処理のための HTTP-HTTPS プロトコル・ハンドラーを表しています。

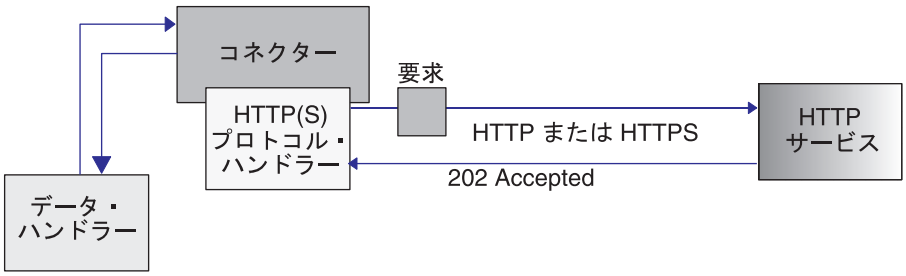


図 17. HTTP-HTTPS プロトコル・ハンドラー: 非同期要求処理

注: このセクションでは、HTTP プロトコル処理についてのみ説明します。

HTTP-HTTPS プロトコル・ハンドラーは、要求ビジネス・オブジェクトのオブジェクト・レベル ASI (`cw_mo_http`) を使用して、プロトコル構成 MO を決定します。HTTP-HTTPS プロトコル・ハンドラーは、HTTP プロトコル構成 MO の Destination 属性を読み取り、ターゲット HTTP サービスの URL を決定します。URL が欠落している、あるいは不完全であれば、プロトコル・ハンドラーはサービス呼び出しで失敗します。 HTTP プロトコル構成 MO とその属性の詳細については、36 ページの『要求処理のための HTTP プロトコル構成 MO』を参照してください。

HTTP-HTTPS プロトコル・ハンドラーは、データ・ハンドラーによって戻される要求メッセージを使用して、HTTP サービスを呼び出します。HTTP プロキシー・コネクターの構成プロパティーが指定されている場合、HTTP-HTTPS プロトコル・ハンドラーは、それに応じた振る舞いをします。応答が戻されると、HTTP-HTTPS プロトコル・ハンドラーはそれを読み取ります。

表 28 は、HTTP-HTTPS プロトコル・ハンドラーが、発信要求メッセージの Charset、MimeType、ContentType、Content-Type ヘッダーを判別するときに使用するルールの優先順位の要約です。

表 28. HTTP-HTTPS プロトコル・ハンドラーによる、アウトバウンド・メッセージの処理ルール

優先順位の順序	**Charset**	**MimeType**	**ContentType**	**Content-Type ヘッダー**
1	プロトコル構成 MO の Content-Type ヘッダー	TLO 属性の MimeType プロパティー	プロトコル構成 MO の Content-Type ヘッダー	プロトコル構成 MO の Content-Type ヘッダー
2	TLO 属性の Charset プロパティー	ContentType にデフォルト設定される		
3	ContentType が `text/*` の場合、ISO-8859-1 にデフォルト設定される。それ以外の場合、Charset は使用されません。			

表 28 は次のことを示しています。

- HTTP-HTTPS プロトコル・ハンドラーは、以下のルールに従って、応答メッセージの Charset を判別します。
  1. Charset 値が要求ビジネス・オブジェクトのプロトコル構成 MO ヘッダーで指定されている場合、その値が使用されます。
  2. 前のステップで Charset を判別できない場合、プロトコル・ハンドラーは、TLO 属性から Charset を抽出しようとします。
  3. 前のステップで説明した操作が失敗すると、Charset を判別するために次の表が使用されます。

表 29. デフォルトの要求処理 Charset

ContentType	**デフォルトの Charset**
`text/*`	ISO-8859-1 詳細については、RFC2616 を参照してください。
`application/*`	デフォルトはなし
その他すべて	デフォルトはなし

  4. 前のステップで Charset が判別すると、その Charset がデータ・ハンドラーに設定されます。
  5. 要求を書き出すのに必要なデータ構造に応じて、データ・ハンドラーが、Stream 配列または Byte 配列の API によって呼び出されます。

- HTTP-HTTPS プロトコル・ハンドラーは、以下のルールに従って、要求の MimeType を判別します。
  1. TLO MimeType 属性。
  2. TLO MimeType 属性がない場合、プロトコル・ハンドラーは、ContentType を使用して MimeType を判別します。
- HTTP-HTTPS プロトコル・ハンドラーは、以下のルールに従って、要求メッセージの ContentType を判別します。
  - Content-Type ヘッダーが要求ビジネス・オブジェクトのプロトコル構成 MO で指定されている場合、ヘッダーのタイプ/サブタイプが ContentType として使用されます。
- HTTP-HTTPS プロトコル・ハンドラーは、以下のルールに従って、要求メッセージの Content-Type ヘッダーを判別します。
  - Content-Type ヘッダーが要求ビジネス・オブジェクトのプロトコル構成 MO で指定されている場合、その値が発信メッセージに使用されます。

表 30 は、ハンドラーが、応答メッセージの Charset、MimeType、ContentType、および Content-Type ヘッダーを判別するときに使用するルールの優先順位の要約です。

表 30. HTTP(s) プロトコル・ハンドラーによる、インバウンド同期応答メッセージの処理ルール

優先順位の順序	Charset	MimeType	ContentType	Content-Type ヘッダー
1	着信 HTTP メッセージの Content-Type ヘッダー値の Charset パラメーター値	要求ビジネス・オブジェクトのプロトコル構成 MO の MessageTransformationMap 子ビジネス・オブジェクト	Content-Type ヘッダー値の、着信 HTTP メッセージのタイプ/サブタイプの値	着信 HTTP メッセージの Content-Type ヘッダー
2	要求ビジネス・オブジェクトのプロトコル構成 MO の MessageTransformationMap 子ビジネス・オブジェクト	要求メッセージの MimeType。ただし、要求と応答の ContentType が一致する場合のみ。		
3	要求メッセージの Charset。ただし、要求と応答の ContentType が一致する場合のみ。	TLO の MimeType プロパティー		
4	TLO の Charset プロパティー	ContentType にデフォルト設定される		
5	Content-Type が `text/*` の場合、ISO-8859-1 にデフォルト設定される。それ以外の場合、Charset は使用されません。			

表 30 は次のことを示しています。

- プロトコル・ハンドラーは、以下のルールに従って、同期応答メッセージの Charset を判別します。
  1. Charset パラメーターが着信応答メッセージの Content-Type ヘッダーに設定されている場合、プロトコル・ハンドラーは、その Charset 値を使用して、データ・ハンドラーに設定します。
  2. 応答メッセージ・ヘッダーに Charset 値がない場合、プロトコル・ハンドラーは、TLO 要求のプロトコル構成 MO の MessageTranformationMap からコラボレーション定義の Charset を読み取ろうとします。
  3. Charset 値が特定の要求の MessageTransformationMap に指定されておらず、応答が要求と同じ ContentType を持つ場合、応答には、要求の Charset が使用されます。
  4. 前のステップで Charset 値が得られない場合、プロトコル・ハンドラーは、TLO の Charset 属性を読み取ろうとします。
  5. 前のステップで説明した方法で Charset 値が取得できない場合で、メッセージ・タイプ ContentType がサブタイプの付いた `text` (`text/xml`、`text/plain` など) のときは、ISO-8859-1 にデフォルト設定されます。それ以外の場合、Charset 値は使用されません。
- プロトコル・ハンドラーは、以下のルールに従って、同期応答メッセージの MimeType を判別します。
  1. プロトコル・ハンドラーは、まず、TLO 要求プロトコル構成 MO の MessageTransformationMap から MimeType を抽出しようとします。特に、プロトコル・ハンドラーは、MTM で正確な ContentType 一致を見つけて、MessageTransformationRule を抽出し、その MimeType プロパティー値を使用しようとします。そうでない場合、プロトコル・ハンドラーは、複数の ContentType に適用される MessageTransformationRule を探します (ContentType が */*)。
  2. MessageTransformationMap を使用して MimeType を判別できない場合、プロトコル・ハンドラーは、要求と応答の ContentTypes が一致するときに限り、応答の MimeType として要求の MimeType を使用します。
  3. これまでのステップを使用して MimeType を抽出できない場合、プロトコル・ハンドラーは、TLO の MimeType 属性を使用します。
  4. これまでのすべてのステップが失敗する場合、プロトコル・ハンドラーは、ContentType を使用して MimeType を設定します。
- ハンドラーは、着信 HTTP メッセージの Content-Type ヘッダーからタイプ/サブタイプを抽出して、ContentType を判別します。

ハンドラーは HTTP プロトコル構成 MO を処理します。HTTP プロトコル構成 MO で渡されるヘッダー値が、要求応答イベントのコンテキストにおいて正しくなるようにするのは、コラボレーションの責任です。ハンドラーは、以下のルールに従って、標準ヘッダーおよびカスタム・プロパティーのデータを取り込みます。

1. ハンドラーは、特殊な属性 (ObjectEventId など) を無視するために、HTTP プロトコル構成 MO の各項目を調べます。
2. 空でない各ヘッダーが発信メッセージに置かれ、追加処理 (Content-Type ヘッダーなど) が行われます。

3. 上記の方法では、ハンドラーは、メッセージに非標準のヘッダーを設定する場合がありますが、メッセージが論理的または文法的に正しいことを保証しないことに注意してください。
4. HTTP プロトコル構成 MO の UserDefinedProperties 属性に 1 つ以上のカスタム・プロパティーがある場合、ハンドラーは、それらを Entity Headers Section (最後のヘッダー・セクション) に追加します。カスタム・プロパティーの詳細については、 37 ページの『要求処理のユーザー定義プロパティー』を参照してください。

注: HTTP プロトコル構成 MO で、Connection、Trailer、Transfer-Encoding、Content-Encoding、Content-Length、Content-MD5、Content-Range のいずれかのヘッダーを指定すると、誤った HTTP メッセージになる可能性が非常に高くなります。

SSL

このセクションでは、コネクターの SSL 機能のインプリメント方法について説明します。背景情報については SSL の資料を参照してください。このセクションでは、SSL テクノロジーについて十分に理解していることを前提としています。

JSSE

コネクターは、JSSE を使用して HTTPS および SSL をサポートします。IBM JSSE はコネクターと共に出荷されます。この機能を使用可能にするには、`java.security` ファイルに次の項目があることを確認してください。このファイルは、コネクターと一緒にインストールされるファイルの 1 つです。

```
security.provider.5=com.ibm.jsse.IBMJSSEProvider
```

`java.security` は、コネクターのインストール先システムの `$ProductDir¥lib¥security` ディレクトリーにあります。コネクターは、JavaProtocolHandlerPackages コネクター・プロパティーの値を使用して、システム・プロパティー java.protocol.handler.pkgs を設定します。コネクターと共に出荷された IBM JSSE の場合、このプロパティーの値は `com.ibm.net.ssl.internal.www.protocol` に設定する必要があることに注意してください。

JavaProtocolHandlerPackages 構成プロパティーでは、この値がデフォルトになります。ただし、システムに値が空でない java.protocol.handler.pkgs システム・プロパティーがある場合、コネクターがそれを上書きするのは JavaProtocolHandlerPackages コネクター・プロパティーも設定されている場合のみです。

コネクターは、初期化中に、JSSE でサポートされる無名の暗号スイートをすべて使用不可にします。

KeyStore および TrustStore

コネクターで SSL を使用するには、鍵ストアとトラストストアをセットアップする必要があります。鍵ストア、証明書、および鍵生成のセットアップ用ツールは提供されていません。これらの作業を完了するには、サード・パーティーのソフトウェア・ツールを使用する必要があります。

SSL プロパティー

コネクター固有の SSL プロパティーとして、以下のプロパティーを指定することができます。

- SSLVersion
- SSLDebug
- KeyStore
- KeyStoreAlias
- KeyStorePassword
- TrustStore
- TrustStorePassword

これらのプロパティーは、コネクター・インスタンスに適用されることに注目してください。コネクターにプラグインされるすべての HTTPS プロトコル・リスナーと、コネクター・インスタンスごとの HTTP-HTTPS プロトコル・ハンドラーにより、同じ SSL プロパティー値のセットが使用されます。HTTPS/SSL セットアップの詳細については、137 ページの『付録 D. HTTPS/SSL の構成』を参照してください。

SSL プロトコル・リスナーおよび HTTPS プロトコル・リスナー

HTTPS プロトコル・リスナーを使用するには、コネクター固有の SSL プロパティーを指定する必要があります。これらのプロパティーに割り当てる値は、以下の SSL 要件を満たしている必要があります。

- **SSLVersion** 使用する SSLVersion が JSSE によってサポートされていることを確認してください。
- **KeyStore** HTTPS プロトコル・リスナーは、SSL 通信のサーバーとして動作するので、鍵ストアを指定する必要があります。リスナーは、SSL->KeyStore 構成プロパティーに指定されている鍵ストアを使用します。このプロパティーの値は、鍵ストア・ファイルの完全パスでなければなりません。鍵ストアにはコネクター用の鍵ペア (秘密鍵と公開鍵) があることを確認してください。秘密鍵の別名を、SSL->KeyStoreAlias プロパティーで指定する必要があります。鍵ストアにアクセスする際に必要なパスワードは、SSL-> KeyStorePassword プロパティーで指定しなければなりません。鍵ストアにアクセスする際に必要なパスワードと秘密鍵 (鍵ストアにある) が同じであることも確認してください。最後に、コネクターのディジタル証明書をクライアントに配布して、コネクターの認証ができるようにする必要があります。
- **TrustStore** HTTPS プロトコル・リスナーでクライアントの認証を行うようにする場合は、クライアント認証をアクティブにしておく必要があります。このためには、SSL ->UseClientAuth プロパティーを `true` に設定します。以下も指定する必要があります。
  - トラストストアのロケーション (SSL->TrustStore 構成プロパティーの値として)
  - トラストストアにアクセスする際に必要なパスワード (SSL-> TrustStorePassword プロパティーの値として)

  トラストストアには、クライアントのディジタル証明書が含まれていることを確認してください。クライアントで使用されるディジタル証明書は、自己署名する

か、または CA から発行されます。トラストストアが CA のルート証明書を信頼すると、JSSE は、その CA によって発行されたすべてのディジタル証明書を認証することになるので注意してください。

HTTPS/SSL セットアップの詳細については、137 ページの『付録 D. HTTPS/SSL の構成』を参照してください。

SSL プロトコル・ハンドラーおよび HTTP-HTTPS プロトコル・ハンドラー

HTTP-HTTPS プロトコル・ハンドラーで SSL を使用する場合は、コネクター固有の SSL プロパティーを指定する必要があります。これらのプロパティーに割り当てる値は、HTTP プロバイダーの以下の HTTPS/SSL 要件を満たしている必要があります。

- **SSLVersion** 使用する SSLVersion がプロバイダーおよび JSSE によってサポートされていることを確認してください。
- **TrustStore** HTTP-HTTPS プロトコル・ハンドラーは SSL 通信ではクライアントとして動作するため、トラストストアをセットアップする必要があります。ハンドラーは、SSL -> Truststore 構成プロパティーに指定されているトラストストアを使用します。このプロパティーの値は、トラストストア・ファイルの完全パスでなければなりません。トラストストアにアクセスする際に必要なパスワードは、SSL -> TrustStorePassword プロパティーに指定しなければなりません。トラストストアには、プロバイダーのディジタル証明書が含まれていることを確認してください。プロバイダーで使用されるディジタル証明書は、自己署名するか、または CA から発行されます。トラストストアが CA のルート証明書を信頼すると、JSSE は、その CA によって発行されたすべてのディジタル証明書を認証することになるので注意してください。
- **KeyStore** HTTP サービス・プロバイダーでクライアント認証が必要な場合は、鍵ストアをセットアップする必要があります。HTTP-HTTPS プロトコル・ハンドラーは、SSL->KeyStore 構成プロパティーに指定されている鍵ストアを使用します。この値は、鍵ストア・ファイルの完全パスでなければなりません。鍵ストアにはコネクター用に構成された鍵ペア (秘密鍵と公開鍵) があることを確認してください。秘密鍵の別名を、SSL->KeyStoreAlias プロパティーに指定する必要があります。鍵ストアにアクセスする際に必要なパスワードは、SSL-> KeyStorePassword プロパティーに指定しなければなりません。最後に、鍵ストアにアクセスする際に必要なパスワードと秘密鍵 (鍵ストアにある) が同じであることを確認してください。認証のためにコネクターのディジタル証明書を HTTP サービス・プロバイダーに配布する必要があります。

HTTPS/SSL セットアップの詳細については、137 ページの『付録 D. HTTPS/SSL の構成』を参照してください。

コネクターの構成

インストーラーを使用してコネクター・ファイルをシステムにインストールした後で、標準およびアプリケーション固有のコネクター構成プロパティーを設定する必要があります。

構成プロパティーの設定

コネクターには、2 種類の構成プロパティーがあります。標準の構成プロパティーと、コネクター固有の構成プロパティーです。コネクターを稼働させる前に、System Manager (SM) を使用して、これらのプロパティーの値を設定する必要があります。

標準構成プロパティー

標準構成プロパティーは、すべてのコネクターによって使用される情報を提供します。これらのプロパティーの説明については、83 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。下の表には、この付録で説明する構成プロパティーに関する、このコネクターに固有の情報が示されています。

プロパティー	説明
CharacterEncoding	このコネクターはこのプロパティーを使用しません。
Locale	このコネクターは国際化に対応していないため、このプロパティーの値は変更できません。現在サポートされているロケールについては、コネクターのリリース情報を参照してください。

このコネクターが統合ブローカーとしてサポートしているのは InterChange Server Express だけなので、このコネクターに関連する構成プロパティーは InterChange Server Express だけに適用されます。

少なくとも、以下の標準コネクター構成プロパティーを設定する必要があります。

- AgentTraceLevel
- ApplicationName
- ControllerTraceLevel
- DeliveryTransport

コネクター固有の構成プロパティー

コネクター固有の構成プロパティーは、実行時にコネクター・エージェントが必要とする情報を提供します。また、コネクター固有のプロパティーを使用することにより、エージェントの再コーディングおよび再ビルドを行わずに、コネクター・エージェント内の静的な情報または論理を変更することができます。

表 31 には、コネクター固有の構成プロパティーがリストされています。プロパティーの説明については、以下の各セクションを参照してください。プロパティーの中には、別のプロパティーを含んでいるものもあります。 + 文字は、プロパティー階層内の項目の位置を示しています。

表 31. コネクター固有の構成プロパティー

名前	指定可能な値	デフォルト値	必須
DataHandlerMetaObjectName	データ・ハンドラーのメタオブジェクト名	MO_DataHandler_ Default	はい
JavaProtocolHandlerPackages	*有効な Java プロトコル・ハンドラー・パッケージ*	com.ibm.net.ssl. internal.www.protocol	なし
ProtocolHandlerFramework	*これは階層プロパティーであり、値はありません。*	なし	なし

表 31. コネクター固有の構成プロパティー (続き)

名前	指定可能な値	デフォルト値	必須
+ProtocolHandlers	これは階層プロパティーであり、値はありません。		なし
++Handler1	これは階層プロパティーです。このプロパティーのサブプロパティーについては、 66 ページの『Handler1』を参照してください。		はい
ProtocolListenerFramework	これは階層プロパティーであり、値はありません。		なし
+WorkerThreadCount	使用可能なリスナー・スレッド数を指定する 1 以上の整数	10	なし
+RequestPoolSize	リソース・プール・サイズを指定する WorkerThreadCount より大きい整数	20	なし
+ProtocolListeners	これは階層プロパティーであり、値はありません。		
++Listener1	固有の名前が指定されたプロトコル・リスナー		はい
+++Protocol	http または https		はい
+++ListenerSpecific	リスナーで固有または必要なプロパティー。 67 ページの『ListenerSpecific』を参照してください。		
ProxyServer	これは階層プロパティーであり、値はありません。		なし
+HttpProxyHost	HTTP プロキシー・サーバーのホスト名		なし
+HttpProxyPort	HTTP プロキシー・サーバーのポート番号	80	なし
+HttpNonProxyHosts	直接接続が必要な HTTP ホスト		なし
+HttpsProxyHost	HTTPS プロキシー・サーバーのホスト名		なし
+HttpsProxyPort	HTTPS プロキシー・サーバーのポート番号	443	なし
+HttpsNonProxyHosts	直接接続が必要な HTTPS ホスト		なし
+SocksProxyHost	Socks プロキシー・サーバーの名前		なし
+SocksProxyPort	Socks プロキシー・サーバーのポート		なし
+HttpProxyUsername	HTTP プロキシー・サーバー・ユーザー名		なし
+HttpProxyPassword	HTTP プロキシー・サーバー・パスワード		なし
+HttpsProxyUsername	HTTPS プロキシー・サーバー・ユーザー名		なし
+HttpsProxyPassword	HTTPS プロキシー・サーバー・パスワード		なし
SSL	これは階層プロパティーであり、値はありません。		なし
+SSLVersion	SSL、SSLv2、SSLv3、TLS、TLSv1	SSL	なし
+SSLDebug	true、false	false	なし
+KeyStoreType	任意の有効な鍵ストア・タイプ	JKS	なし
+KeyStore	KeyStore ファイルへのパス		なし

表 31. コネクター固有の構成プロパティー (続き)

名前	指定可能な値	デフォルト値	必須
+KeyStorePassword	*KeyStore* における秘密鍵のパスワード		なし
+KeyStoreAlias	*KeyStore* における鍵ペアの別名		なし
+TrustStore	*TrustStore* ファイルへのパス		なし
+TrustStorePassword	*TrustStore* のパスワード		なし
+UseClientAuth	`true`、`false`	`false`	なし

DataHandlerMetaObjectName: これは、データ・ハンドラーが構成プロパティーを設定するために使用する、メタオブジェクトの名前です。

デフォルト値は `MO_DataHandler_Default` です。

JavaProtocolHandlerPackages: このプロパティーの値は、Java プロトコル・ハンドラー・パッケージを指定します。コネクターは、このプロパティーの値を使用して、システム・プロパティー java.protocol.handler.pkgs を設定します。

デフォルトは `com.ibm.net.ssl.internal.www.protocol` です。

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

デフォルト値はありません。

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

デフォルト値はありません。

Handler1: HTTP-HTTPS プロトコル・ハンドラーの名前。これは階層プロパティーです。リスナーとは異なり、プロトコル・ハンドラーは重複しない場合もあるので、それぞれのプロトコルごとにハンドラーは 1 つだけしかないことがあります。表 32 は HTTP-HTTPS プロトコル・ハンドラーのサブプロパティーを表しています。 + 文字は、プロパティー階層内の項目の位置を示しています。

表 32. HTTP-HTTPS プロトコル・ハンドラー構成プロパティー

名前	指定可能な値	デフォルト値	必須
++HTTPHTTPSHandler	これは階層プロパティーであり、値はありません。		はい
+++Protocol	ハンドラーがインプリメントしようとしているプロトコルの種類。*HTTP* および *HTTPS* の場合の値は `http` です。 **注:** このプロパティーの値を指定しなければ、コネクターはプロトコル・ハンドラーを初期化しません。	`http`	はい

表 32. HTTP-HTTPS プロトコル・ハンドラー構成プロパティー (続き)

名前	指定可能な値	デフォルト値	必須
+++HTTPReadTimeout	*リモート・ホストから読み取りを行う際のタイムアウト・インターバル (ミリ秒単位) を指定する、HTTP 固有のプロパティー。このプロパティーが指定されていない場合、または 0 に設定されている場合、HTTP プロトコル・ハンドラーは、リモート・ホストからの読み取りを行っている間、いつまでもブロックします。*	0	なし

ProtocolListenerFramework: プロトコル・リスナー・フレームワークは、このプロパティーを使用してプロトコル・リスナーをロードします。これは階層プロパティーであり、値はありません。

WorkerThreadCount: このプロパティー (これは、1 以上の整数でなければなりません) は、プロトコル・リスナー・フレームワークで使用できるプロトコル・リスナー・ワーカー・スレッドの数を設定します。詳細については、48 ページの『プロトコル・リスナー』を参照してください。デフォルト値は `10` です。

RequestPoolSize: このプロパティー (これは、WorkerThreadCount より大きい整数でなければなりません) は、プロトコル・リスナー・フレームワークのリソース・プール・サイズを設定します。このフレームワークは、最大で、WorkerThreadCount 要求と RequestPoolSize 要求の合計数を並行して処理することができます。

デフォルト値は `20` です。

ProtocolListeners: これは階層プロパティーであり、値はありません。このプロパティーの第 1 レベルの子は、それぞれ、別個のプロトコル・リスナーを表します。

Listener1: プロトコル・リスナーの名前。複数のプロトコル・リスナーが存在する場合があります。これは階層プロパティーです。このプロパティーのインスタンスを複数作成して、固有の名前を持つ、追加のリスナーを作成することができます。このようにする場合、リスナー固有のプロパティーは変更できますが、プロトコル・プロパティーは変更できません。複数のリスナーの名前は固有でなければなりません。指定可能な名前 (値ではありません) は、`HTTPListener1` または `HTTPSListener1` です。

Protocol: このプロパティーは、このリスナーがインプリメントしようとしているプロトコルを指定します。指定可能な値は、`http` または `https` です。

注: このプロパティーの値を指定しなければ、コネクターはプロトコル・リスナーを初期化しません。

ListenerSpecific: リスナー固有のプロパティーは、指定されたプロトコル・リスナーに固有の、あるいは指定されたプロトコル・リスナーにとって必須のプロパティーです。例えば、HTTP リスナーにはリスナー固有のプロパティー Port が設定されますが、これは、リスナーが要求をモニターするポート番号を表します。表 33 は、HTTP/HTTPS リスナーに固有のプロパティーを要約したものです。 + 文字は、プロパティー階層内の項目の位置を示しています。

表 33. HTTP および HTTPS プロトコル・リスナーに固有の構成プロパティー

名前	指定可能な値	デフォルト値	必須
+++HTTPListener1	*HTTP プロトコル・リスナーの固有の名前。これは、ProtocolListenerFramework -> ProtocolListeners 階層プロパティーの子です。リスナーは複数存在する場合があります。このプロパティーおよびその階層の別のインスタンスを作成することにより、追加の HTTP リスナーをプラグインできます。*		はい
++++Protocol	`http` (HTTP プロトコル・リスナーの場合) `https` (HTTPS プロトコル・リスナーの場合) **注:** このプロパティーの値を指定しなければ、コネクターはプロトコル・リスナーを初期化しません。		はい
++++BOPrefix	*このプロパティーの値はデータ・ハンドラーに渡されます。*		なし
++++Host	*リスナーは、このプロパティーの値によって指定された IP アドレスで要求を listen します。*Host *が指定されない場合、デフォルト値として localhost が使用されます。リスナーを実行しているマシンのホスト名 (DNS 名) または IP アドレスのどちらを指定しても構いません。 1 台のマシンで複数の IP アドレスまたは複数の名前を持つこともできます。*	`localhost`	なし
++++Port	*リスナーが要求を listen するポート。ポートが指定されていない場合、ポートのデフォルトとして、HTTP の場合は* `80`*、HTTPS の場合は* `443` *が使用されます。コネクター内でリスナーを複製した場合、Host プロパティーと Port プロパティーの組み合わせは固有にします。そうしないと、リスナーは、要求を受け入れるためにポートにバインドすることができなくなる可能性があります。*	HTTP リスナーの場合は `80` HTTPS リスナーの場合は `443`	なし
++++SocketQueueLength	*着信接続要求のためのキュー (ソケット・キュー) の長さ。ホストに接続を拒否されずに、一度に保管できる着信接続の数を指定します。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。*	`5`	なし
++++RequestWaitTimeout	*要求の到着を待つ際にリスナー・スレッドがホストおよびポートでブロックを行う時間間隔を、ミリ秒単位で指定します。この間隔以内に 要求を受け取った場合、リスナーはその要求を処理します。それ以外の場合、リスナー・スレッドは、コネクター・シャットダウン・フラグが設定されているかどうかを調べます。このフラグが設定されている場合、コネクターは終了します。このフラグが設定されていない場合、RequestWaitTimeout 間隔の間、ブロックが継続します。このプロパティーを 0 に設定した場合、ブロックはいつまでも継続します。このプロパティーを指定しないと、デフォルトとして 60000 ミリ秒が使用されます。*	`60000` (ms)	なし
++++HTTPReadTimeout	*クライアントからの要求を読み取っているときにリスナーがブロックされる時間間隔を、ミリ秒単位で指定します。このパラメーターを 0 に設定すると、リスナーは、要求メッセージ全体を受け取るまで、いつまでもブロックされます。*	`0`	なし

表 33. *HTTP および HTTPS プロトコル・リスナーに固有の構成プロパティー (続き)*

名前	指定可能な値	デフォルト値	必須
++++HttpAsyncResponseCode	*リスナーに対する非同期要求の HTTP 応答コードです。* 200 (OK) 202 (ACCEPTED)	202 (ACCEPTED)	なし
++++URLsConfiguration	*これは階層プロパティーであり、値はありません。このリスナーによってサポートされる URL の 1 つ以上の構成が含まれます。オプションとして、MIME タイプおよび文字セット値が含まれます。これは、* ProtocolListenerFramework->ProtocolListeners->HTTPListener1 *階層プロパティーの子プロパティーであることに注意してください。このプロパティーが指定されないと、リスナーはデフォルト値を想定します。*	ContextPath: / 使用可能: true データ・ハンドラー MimeType: 要求の ContentType に等しい Charset: なし。 詳細については、 49 ページの『HTTP および HTTPS プロトコル・リスナー処理』を参照してください。	なし
+++++URL1	これは階層プロパティーであり、値はありません。その子は、このリスナーによってサポートされる *URL の名前を提供します。サポートされる URL は複数存在する場合があります。このプロパティーとその階層のクローンを作成して、追加 URL を接続できることに注意してください。*		なし
++++++ContextPath	*リスナーが受け取る HTTP 要求の URI。これは、URLsConfiguration プロパティーのそれぞれの ContextPath 値において固有の値でなければなりません。そうでない場合、コネクターはエラーをログに記録し、始動に失敗します。ContextPath の値については、大文字小文字の区別が行われます。ただし、大文字小文字が区別されないプロトコル、ホスト名、およびポートを含めることができます。プロトコルを ContextPath に指定する場合は、http にする必要があります。ホストを指定する場合には、Host リスナー・プロパティーの値と同じにする必要があります。ポートを指定する場合には、Port リスナー・プロパティーの値と同じにする必要があります。*		なし
++++++Enabled	*このプロパティーの値は、コネクターで親 URL 階層プロパティーが使用可能かどうかを判別します。*	True	なし
++++++TransformationRules	*これは階層プロパティーであり、値はありません。1 つ以上の変換ルールを保持します。*		
+++++++TransformationRule1	これは階層プロパティーであり、値はありません。*変換ルールを保持します。*		なし

表 33. HTTP および HTTPS プロトコル・リスナーに固有の構成プロパティー (続き)

名前	指定可能な値	デフォルト値	必須
++++++++ContentType	このプロパティーの値は、特別な処理 (データ・ハンドラー MIME タイプまたは文字セット) を適用する必要のある着信要求の *ContentType* を指定します。*ContentType* が *TransformationRule*N 階層プロパティーによって指定されない場合、コネクターは警告メッセージをログに記録し、*TransformationRule*N プロパティーを無視します。このプロパティーに特殊値 */* を指定すると、プロトコル・リスナーは、このルールを任意の *ContentType* に適用できます。*ContentType* を共用する同一コンテキスト・パスで複数のルールを見つけた場合、リスナーはエラーをログに記録し、初期化に失敗することに注意してください。		なし
++++++++MimeType	指定された *ContentType* の要求を処理するデータ・ハンドラーを呼び出すときに使用する *MIME* タイプ。		なし
++++++++Charset	指定された *ContentType* の要求をビジネス・オブジェクトに変換するときに使用する文字セット。		なし

図 18 は、Connector Configurator Express で表示されたプロパティーを表しています。

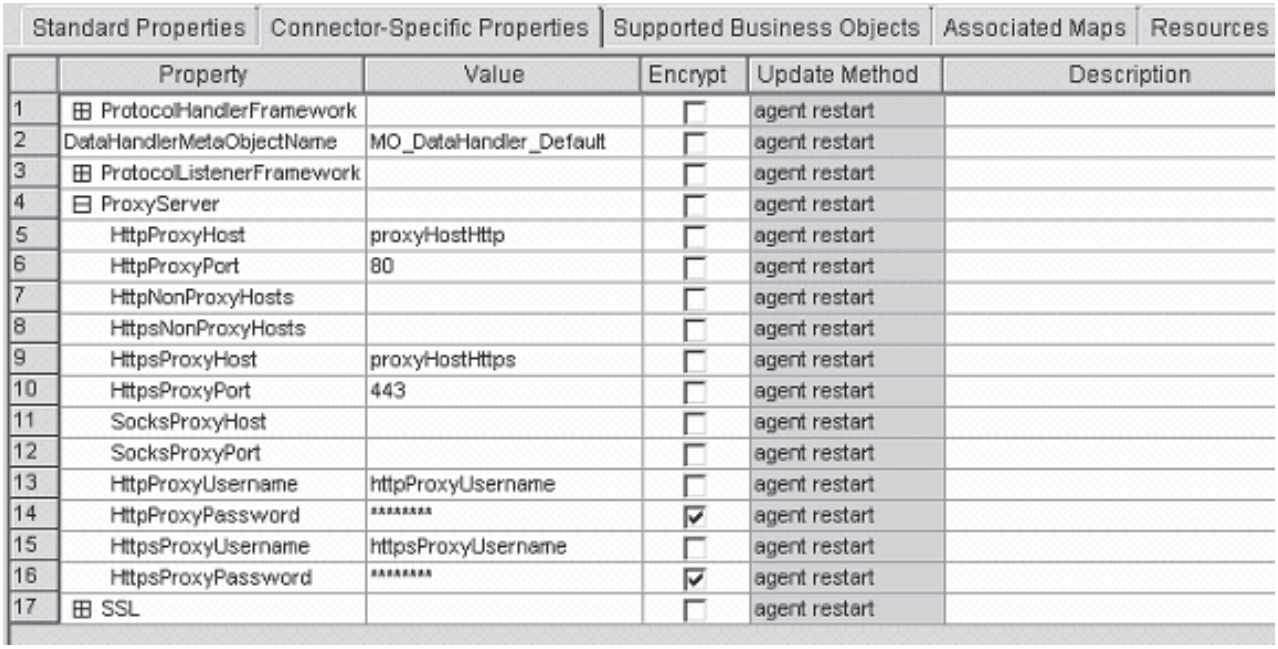
Standard Properties | Connector-Specific Properties | Supported Business Objects | Associated Maps | Resources

	Property	Value	Encrypt	Update Method	Description
1	⊞ ProtocolHandlerFramework		☐	agent restart	
2	DataHandlerMetaObjectName	MO_DataHandler_Default	☐	agent restart	
3	⊟ ProtocolListenerFramework		☐	agent restart	
4	WorkerThreadCount	10	☐	agent restart	
5	RequestPoolSize	20	☐	agent restart	
6	⊟ ProtocolListeners		☐	agent restart	
7	⊞ HTTPListener1		☐	agent restart	
8	⊟ HTTPSListener1		☐	agent restart	
9	Protocol	https	☐	agent restart	
10	Host	localhost	☐	agent restart	
11	Port	8443	☐	agent restart	
12	SocketQueueLength	5	☐	agent restart	
13	HTTPReadTimeout	0	☐	agent restart	
14	RequestWaitTimeout	60000	☐	agent restart	
15	BOPrefix		☐	agent restart	
16	⊞ URLsConfiguration		☐	agent restart	
17	⊞ ProxyServer		☐	agent restart	
18	⊞ SSL		☐	agent restart	

図 18. HTTP(S) プロトコル・リスナー・プロパティー

ProxyServer: ネットワークがプロキシー・サーバーを使用する場合には、このプロパティー以下の値を構成してください。これは階層プロパティーであり、値はありません。このプロパティー以下で指定された値は、HTTP-HTTPS プロトコル・ハンドラーによって使用されます。

図 19 は、Connector Configurator Express で表示された ProxyServer プロパティーを表しています。



	Property	Value	Encrypt	Update Method	Description
1	⊞ ProtocolHandlerFramework		☐	agent restart	
2	DataHandlerMetaObjectName	MO_DataHandler_Default	☐	agent restart	
3	⊞ ProtocolListenerFramework		☐	agent restart	
4	⊟ ProxyServer		☐	agent restart	
5	HttpProxyHost	proxyHostHttp	☐	agent restart	
6	HttpProxyPort	80	☐	agent restart	
7	HttpNonProxyHosts		☐	agent restart	
8	HttpsNonProxyHosts		☐	agent restart	
9	HttpsProxyHost	proxyHostHttps	☐	agent restart	
10	HttpsProxyPort	443	☐	agent restart	
11	SocksProxyHost		☐	agent restart	
12	SocksProxyPort		☐	agent restart	
13	HttpProxyUsername	httpProxyUsername	☐	agent restart	
14	HttpProxyPassword	********	☑	agent restart	
15	HttpsProxyUsername	httpsProxyUsername	☐	agent restart	
16	HttpsProxyPassword	********	☑	agent restart	
17	⊞ SSL		☐	agent restart	

図 19. ProxyServer プロパティー

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

デフォルト値はありません。

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

デフォルト値は 80 です。

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

デフォルト値はありません。

HttpsProxyHost: HTTPS プロキシー・サーバーの場合のホスト名。

デフォルト値はありません。

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

デフォルト値は 443 です。

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

デフォルト値はありません。

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

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

デフォルト値はありません。

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

デフォルト値はありません。

HttpProxyUsername: HTTP プロキシー・サーバーのユーザー名。要求の宛先が HTTP URL のときに、ProxyServer ->HttpProxyUsername を指定する場合、HTTP-HTTPS プロトコル・ハンドラーは、プロキシーを認証するときに Proxy-Authorization ヘッダーを作成します。ハンドラーは、認証に CONNECT メソッドを使用します。

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

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

デフォルト値はありません。

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

デフォルト値はありません。

HttpsProxyUsername: HTTPS プロキシー・サーバーのユーザー名。要求の宛先が HTTPS URL のときに、ProxyServer ->HttpsProxyUsername を指定する場合、HTTP-HTTPS プロトコル・ハンドラーは、プロキシーの認証用に Proxy-Authorization ヘッダーを作成します。ハンドラーは、HttpsProxyUsername と HttpsProxyPassword の構成プロパティー値 (コロン (:) によって区切られた) を連結し、Base64 でエンコードされたストリングを作成します。

デフォルト値はありません。

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

デフォルト値はありません。

SSL: コネクターのための SSL を構成するためには、このプロパティー以下の値を指定してください。これは階層プロパティーであり、値はありません。

図 20 は、Connector Configurator Express で表示された SSL プロパティーを表しています。

Standard Properties | Connector-Specific Properties | Supported Business Objects | Associated Maps | Resources

	Property	Value	Encrypt	Update Method	Description
1	⊞ ProtocolHandlerFramework		☐	agent restart	
2	DataHandlerMetaObjectName	MO_DataHandler_Default	☐	agent restart	
3	⊞ ProtocolListenerFramework		☐	agent restart	
4	⊞ ProxyServer		☐	agent restart	
5	⊟ SSL		☐	agent restart	
6	SSLVersion	SSL	☐	agent restart	
7	SSLDebug	False	☐	agent restart	
8	KeyStoreType	JKS	☐	agent restart	
9	KeyStore		☐	agent restart	
10	KeyStorePassword		☐	agent restart	
11	KeyStoreAlias		☐	agent restart	
12	TrustStore		☐	agent restart	
13	TrustStorePassword		☐	agent restart	
14	UseClientAuth	False	☐	agent restart	

図 20. SSL プロパティー

SSLVersion: コネクターによって使用される SSL のバージョン。詳しくは、IBM JSSE 資料に記載された、サポートされる SSL バージョンに関する説明を参照してください。

デフォルト値は `SSL` です。

SSLDebug: このプロパティーの値が `true` に設定されている場合、コネクターは `javax.net.debug` システム・プロパティーの値を `true` に設定します。IBM JSSE はこのプロパティーを使用して、トレース機能をオンにします。詳しくは IBM JSSE の資料を参照してください。

デフォルト値は `false` です。

KeyStoreType: このプロパティーの値は、KeyStore および TrustStore のタイプを表します。詳しくは、IBM JSSE 資料に記載された、有効な鍵ストア・タイプに関する説明を参照してください。

デフォルト値は `JKS` です。

KeyStore: このプロパティーでは、鍵ストア・ファイルへの完全パスを指定します。KeyStore または KeyStoreAlias プロパティー、あるいはその両方が指定されていない場合には、KeyStorePassword、KeyStoreAlias、TrustStore、および TrustStorePassword の各プロパティーは無視されます。コネクターは、このプロパテ

ィーで指定されたパスを使用して鍵ストアをロードすることができない場合、始動に失敗します。このパスは、鍵ストア・ファイルの完全パスでなければなりません。

デフォルト値はありません。

KeyStorePassword: このプロパティーでは、Keystore 内の秘密鍵のパスワードを指定します。

デフォルト値はありません。

KeyStoreAlias: このプロパティーでは、KeyStore 内の鍵ペアの別名を指定します。HTTPS リスナーは、KeyStore からこの秘密鍵を入手して使用します。また、HTTP-HTTPS プロトコル・ハンドラーは、クライアント認証を必要とする HTTPS サービスを呼び出すときに、KeyStore からこの別名を入手して使用します。このプロパティーは、有効な JSSE 別名に設定しなければなりません。

デフォルト値はありません。

TrustStore: このプロパティーでは、TrustStore への完全パスを指定します。TrustStore は、コネクターに信頼される証明書を保管するために使用されます。TrustStore のタイプは KeyStore と同じでなければなりません。TrustStore ファイルの完全パスを指定しなければなりません。

デフォルト値はありません。

TrustStorePassword: このプロパティーは、Truststore のパスワードを指定します。

デフォルト値はありません。

UseClientAuth: このプロパティーでは、SSL クライアント認証を使用するかどうかを指定します。このプロパティーが `true` に設定されている場合、HTTPS リスナーはクライアント認証を使用します。

デフォルト値は `false` です。

複数のプロトコル・リスナーの作成

プロトコル・リスナーのインスタンスを複数作成することができます。プロトコル・リスナーは、ProtocolListenerFramework -> ProtocolListeners コネクター・プロパティーの子プロパティーとして構成されます。(ProtocolListenerFramework -> ProtocolListeners の) それぞれの子プロパティーは、コネクターのプロトコル・リスナーを明確に識別します。したがって、ProtocolListeners プロパティーの下で、新しい子プロパティーを構成することにより、追加のプロトコル・リスナーを作成することができます。新規に作成したリスナー・プロパティーの子プロパティーをすべて指定していることを、確認してください。各リスナーの名前は固有でなければなりません。ただし、リスナーのプロトコル・プロパティー (`http` または `https`) は変更しません。これらは、リスナーの複数インスタンス用に同じ名前で残ります。

注: このプロトコル・プロパティーはスイッチの役目をしているため非常に重要です。リスナーまたはハンドラーを使用したくない場合は、このプロパティーを空のままにします。

HTTP リスナーまたは HTTPS リスナーの複数インスタンスを作成している場合は、それぞれのインスタンスに別々のポートおよびホスト・プロパティーを必ず指定してください。

1 つのハンドラーの複数のインスタンスは作成できません。プロトコルごとにハンドラーは 1 つしか存在できません。

始動時のコネクター

コネクターを始動すると、`init()` メソッドは、System Manager の Connector Configurator Express を使用して設定された構成プロパティーを読み取ります。正常に機能させるために、絶対にコネクターのポーリングを使用不可にしないでください (コネクターのポーリングはデフォルトで使用可能になっています)。この際にどのようなことが行われるのかについて、以下のセクションで説明します。

プロキシーのセットアップ

コネクター固有の ProxyServer プロパティーを指定すると、コネクターはプロキシー・システム・プロパティーをセットアップします。プロキシー・サーバーは、要求処理の場合のみ、HTTP-HTTPS プロトコル・ハンドラーとともに使用されます。コネクターはまた、セットアップしたそれぞれのシステム・プロパティーをトレースします。ProxyServer プロパティーの詳細については、64 ページの『コネクター固有の構成プロパティー』を参照してください。

プロトコル・リスナー・フレームワークの初期化

コネクターは、始動時にプロトコル・リスナー・フレームワークのインスタンスを作成し、それを初期化します。このフレームワークがコネクター固有の ProtocolListenerFramework プロパティーを読み取り、さらにコネクターがコネクター・プロパティー WorkerThreads および RequestPoolSize の値を読み取ります。ProtocolListenerFramework プロパティーが指定されていないかあるいは欠落している場合、コネクターはクライアントから要求を受け取ることができず、警告をログに記録します。

コネクターは、次に ProtocolListenerFramework -> ProtocolListeners プロパティーを読み取ります。ProtocolListeners プロパティーのすべての第 1 レベル・プロパティーは、プロトコル・リスナーを表します。プロトコル・リスナー・フレームワークは、それぞれのリスナーのロードと初期化を試み、それらをトレースします。永続的イベントに対応している場合には、リスナーはイベント・リカバリーを試みます。

プロトコル・ハンドラー・フレームワークの初期化

コネクターは、コネクター固有の ProtocolHandlerFramework プロパティーを読み取り、プロトコル・ハンドラー・フレームワークのインスタンスを作成してそれを初期化します。このプロパティーが欠落しているか、あるいは正しく設定されていない場合、コネクターは要求を処理することができず、警告をログに記録します。次に、コネクターはすべての ProtocolHandlerFramework -> ProtocolHandlers プロパティー (これは、プロトコル・ハンドラーに対応します) を読み取り、それらのプロパティーのロード、初期化、およびトレースを試みます。プロトコル・ハンドラーは

コネクターの初期化中にロードされ、コラボレーションがサービス要求を行うときにはインスタンス化されません。プロトコル・ハンドラーはマルチスレッドに対応しています。

ロギング

コネクターは、以下の場合にログに警告を記録します。

- ProtocolListenerFramework プロパティーが指定されていない場合。コネクターは、イベント通知を行えないことを警告します。
- ProtocolHandlerFramework プロパティーが指定されていない場合。コネクターは、(コラボレーション) 要求処理を行えないことを警告します。

トレース

トレースは、コネクターの振る舞いを綿密にたどるために使用することのできる、オプションのデバッグ機能です。デフォルトでは、トレース・メッセージは STDOUT に書き込まれるようになっています。トレース・メッセージの構成に関する詳細については、コネクター構成プロパティーを参照してください。

コネクターのトレース・レベルは次のとおりです。

レベル 0　このレベルは、コネクターのバージョンを識別するトレース・メッセージに使用されます。

レベル 1　pollForEvents メソッドが呼び出されるたびにトレースを行います。InterChange Server Express への送達のためにリスナーによって作成された TLO 名をトレースします。要求ビジネス・オブジェクト名、およびそれに対応する TLO 内の属性名をトレースします。

レベル 2　ビジネス・オブジェクトが `gotApplEvent()` または `executeCollaboration()` から InterChange Server Express に送付されるたびに記録されるトレース・メッセージに使用します。また、どのプロトコル・ハンドラーが要求を処理しているのかもトレースします。

レベル 3　処理中のビジネス・オブジェクトの ASI をトレースします。処理中のビジネス・オブジェクトの属性をトレースします。イベント通知中に要求ビジネス・オブジェクトの TLO をトレースします。データ・ハンドラーによって戻されたビジネス・オブジェクトをトレースします。

レベル 4　以下のものと関連したトランスポート・ヘッダーをトレースします。

- プロトコル・リスナーによってトランスポートから検索された要求メッセージ
- プロトコル・リスナーによってクライアントに送信された応答メッセージ

スレッドの作成、処理されるすべての ASI、および重要な機能を持つすべての入り口および出口をトレースします。

レベル 5　以下のものをトレースします。

- それぞれの重要なメソッドごとの入り口および出口
- 構成に固有のすべてのプロパティー
- それぞれのプロトコル・リスナーのロード
- プロトコル・リスナーによってトランスポートから検索された要求メッセージ
- プロトコル・リスナーによってトランスポートでクライアントに送信された応答メッセージ
- それぞれのプロトコル・ハンドラーのロード
- データ・ハンドラーによって戻されたメッセージ
- コラボレーションに送られた TLO のビジネス・オブジェクト・ダンプ
- データ・ハンドラーによって戻されたビジネス・オブジェクトのダンプ

第 5 章 トラブルシューティング

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

始動時の問題

問題	考えられる処置/説明
アルゴリズムがサポートされていない/アルゴリズム「SSL」が使用できない。(Algorithm Not Supported/Algorithm 'SSL' not available)	このエラーは、Connector Configurator Express で指定された SSL バージョンが JSSE プロバイダーによってサポートされない場合に発生します。ソリューション: サポートされる SSL バージョンについての JSSE プロバイダーの資料を調べます。IBM JSSE の場合は、*ProductDir*/lib/security ディレクトリー内の java.security ファイルに、以下の項目が含まれていることを確認します。 `security.provider.<number>=com.ibm.jsse.IBMJSSEProvider` ここで、`<number>` は、セキュリティー・プロバイダーをロードする際の優先順位です。
鍵ストアのロード中にエラーが発生する。鍵ストア・ファイル・パス *"<path>"* の指定が間違っている。鍵ストアが見つからない。(Error loading keystore:Keystore file path:*"<path>"* incorrectly specified:KeyStore not found)	このエラーは、鍵ストアおよび/またはトラストストア・ファイルに間違ったパスを指定した場合に発生します。ソリューション: Connector Configurator Express 内の「SSL」->「鍵ストア (KeyStore)」プロパティーで指定された鍵ストア・ファイル・パスを調べます。また、トラストストアを使用している場合は、Connector Configurator Express 内の「SSL」->「トラストストア (TrustStore)」プロパティーで指定されたトラストストア・ファイル・パスを調べます。
KeyManagementError。鍵ストアが改ざんされ、鍵管理エラーが出る。(KeyManagementError: KeyStore is tampered with, KeyManagement error)	このエラーは、鍵ストアおよび/またはトラストストアが改ざんされたか、あるいは破壊された場合に発生します。また、このエラーは、パスワードに間違った値を指定した場合にも発生します。ソリューション: 鍵ストアが改ざんされていないことを確認します。鍵ストアの再作成を試みます。また、正しいパスワードを「SSL」->「KeyStorePassword」および「SSL」->「TrustStorePassword」コネクター・プロパティーに入力したことを確認します。
鍵ストアから証明書のロード中にエラーが発生する。(Error loading certificates from keystore)	このエラーは、証明書および/または鍵ストア、トラストストアが改ざんされた場合に発生します。また、このエラーは、パスワードに間違った値を指定した場合にも発生します。ソリューション: 証明書、鍵ストア、またはトラストストアが改ざんされているかどうかを調べます。また、正しいパスワードを「SSL」->「KeyStorePassword」および「SSL」->「TruststorePassword」コネクター・プロパティーに入力したことを確認します。

問題	考えられる処置/説明
サーバー・ソケットの作成中にエラーが発生し、終了する: エラー。(Error creating the server socket, terminating: error)	このエラーは、HTTP または HTTPS プロトコル・リスナーが、コネクター・プロパティーで指定されたポートにバインドできない場合に発生します。ソリューション: すべての HTTP および HTTPS プロトコル・リスナーに対して指定したポートを調べます。同一のポートが複数のリスナーに対して指定されている場合は、リスナーの 1 つだけが始動できます。さらに、そのポート上で実行中のほかのサービスがあるかどうか調べます。そのポート上で実行中のほかのサービスがある場合には、別のポートをプロトコル・リスナー用に選択することもできます。
KeyManagementError:UnrecoverableKeyException、鍵が回復できなかった。 (KeyManagementError:UnrecoverableKeyException, Keys could not be recovered)	このエラーは、鍵ストアまたはトラストストアを使用できない場合に発生します。ソリューション: 鍵ストアを新規に作成します。
SSL ハンドシェーク例外: 不明なCA。(SSL Handshake Exception: Unknown CA)	これは、CA 証明書がトラストストア内にない場合に発生します。ソリューション: CA の証明書のほか、その自己署名証明書がトラストストア内に存在しているかどうかを調べます。さらに、証明書の DN にホスト名 (なるべく IP アドレス) が含まれていることを確認します。
ログ・ファイル内の過剰な JSSE ロギングに気付く。	基盤となる JSSE の詳細をすべてコンソール上に表示する必要がない場合は、Connector Configurator Express 内の「SSL」->「SSLDebug」プロパティーの値を `false` に設定します。
プロトコル・リスナーを指定したが、そのリスナーが初期化されないで、以下の警告メッセージがコネクターに表示される。 `Skipping Protocol Listener Property Set "SOME_LISTENER_NAME" with protocol property "": unable to determine the protocol listener class.]`	コネクターが、プロトコル・リスナーの「プロトコル (Protocol)」プロパティーに対して有効な値を取り出せませんでした。有効な値は、`http` または `https` です。ソリューション: これはエラー状態ではありません。しかし、コネクターにこのリスナーを使用させる場合は、有効な値を「プロトコル (Protocol)」プロパティーに指定します。
プロトコル・ハンドラーを指定したが、そのハンドラーが初期化されないで、以下の警告メッセージがコネクターに表示される。 `Unable to determine the type of the handler; skipping initializing of current handler. Handler property details: Name: <Handler Name>; Value: Name: Protocol; Value: Name: ResponseWaitTimeout; Value: Name: ReplyToQueue; Value: .]`	コネクターが、ハンドラーの「プロトコル (Protocol)」プロパティーに対して有効な値を取り出せませんでした。有効な値は、`http` または `https` です。ソリューション: これはエラー状態ではありません。しかし、コネクターにこのハンドラーを使用させる場合は、有効な値を「プロトコル (Protocol)」プロパティーに指定します。

ランタイム・エラー

問題	考えられる処置/説明
`HTTP` 応答の解析中にエラーが発生する。`HTTP` 応答ヘッダーの読み取り中にストリームの終わりに到達した。(`Error parsing HTTP response:Reached end of stream while reading HTTP response header`)	このエラーは、コネクターが HTTP サービスを呼び出すときに発生します。これは、ターゲット HTTP サービスが、間違った HTTP 応答を送信したために発生します。ソリューション: ターゲット HTTP サービスのアドレスが正しいことを確認します。

問題	考えられる処置/説明
指定された URL でエラーが発生した。ホストおよびポートの詳細を抽出できない。宛先が誤っている。 <destination URL>(Error in the url mentioned , unable to extract host and port details ,destination is wrong <destination URL>)	このエラーは、コネクターが HTTP サービスを呼び出すときに発生します。これは、HTTP サービスに間違ったエンドポイント・アドレスを指定したために発生します。ソリューション: HTTP サービスに正しいアドレスを指定したことを確認します。
動詞 <*Verb*> でイベント・ビジネス・オブジェクト <*BO Name*> をブローカーに送信中に失敗した。(Failure in sending event business object <*BO Name*> with verb <*Verb*> to the broker.)実行状況「-1」と以下のエラー・メッセージを受け取った。(Received execution status ″-1″ and error message:)	このエラーが発生するのは、コネクターによるイベントの同期送信先のコラボレーションが存在していないか、あるいはビジネス・オブジェクトの動詞を受け入れないかのいずれかであるため、統合ブローカーがそのイベントの処理に失敗する場合です。ソリューション: イベント通知のために TLO を使用している場合は、TLO の ws_collab オブジェクト・レベルの ASI を調べます。(TLO の名前は、エラー・メッセージに示されます。) ws_collab ASI の値を調べます。このコラボレーションが存在して、稼働していることを確認します。ws_mode BO レベルの ASI が synch に設定されている場合は、ws_collab ASI が必要です。ws_verb オブジェクト・レベルの ASI の値を調べます。ws_collab ASI で指定したコラボレーションを、ws_verb ASI で指定した動詞によって起動できることを確認します。このコラボレーションが存在して、稼働していることを確認します。
MapException: コネクター・コントローラー HTTPConnector のマップ・ビジネス・オブジェクト <BO Name> へのマップが見つかりません。(MapException: Unable to find the map to map business objects <BO Name> for the connector controller HTTPConnector)	
要求の要求ビジネス・オブジェクトへの変換に失敗する。障害: (Failed to transform a request into a request business object. Fault:)	このエラーは、コネクターが統合ブローカーに送信しようとしているビジネス・オブジェクトの動詞を判別できない場合に、イベント通知の際に発生します。ソリューション: この TLO に対して ws_verb オブジェクト・レベルの ASI を指定したことを確認します。動詞をこの ASI の値として指定します。
要求オブジェクトの生成中に障害が発生しました。要求ビジネス・オブジェクトに動詞を設定できませんでした。(Failure in generating request object - no verb could be set on the request bo)	

付録 A. コネクターの標準構成プロパティー

この付録では、WebSphere Business Integration 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

Connector Configurator Express からコネクター・プロパティーを構成します。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 に関する付録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、新規の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、新規の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、85 ページの表 34 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 34 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 34 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 34. 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 MQ または IDL で、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS の場合のみ有効です。
AgentTraceLevel	0 から 5	0	ICS では動的、その他の場合はコンポーネント再始動	

表 34. 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
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	ICS	ICS	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> に設定されて、BrokerType の値が ICS の場合のみ有効です。
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 ICS の場合のみ有効です。
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 ICS の場合のみ有効です。
ControllerTraceLevel	0 から 5	0	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定されて、BrokerType の値が ICS の場合のみ有効です。
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。
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。

表 34. 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
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 ICS 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	jms.FactoryClassName の値が IBM の場合は、crossworlds.queue.manager を使用します。	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 ICS の場合のみ有効です。
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 ICS の場合のみ有効です。
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 ICS の場合のみ有効です。
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 ICS の場合のみ有効です。
ListenerConcurrency	1 から 100	1	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が MQ の場合のみ有効です。

表 34. 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、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 ICS の場合のみ有効です。
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 ICS の場合のみ有効です。
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 ICS の場合のみ有効です。
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 ICS の場合のみ有効です。
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 ICS の場合のみ有効です。
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 ICS の場合は動的。そうでない場合は、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 34. 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 ICS の場合は <REMOTE>。それ以外の場合は任意の有効なローカル・ディレクトリー。	ICS 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	ICS 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	ICS 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 ICS の場合、このプロパティーは無効です。

表 34. 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 ICS 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、統合ブローカーがコネクターへ管理メッセージを送信するときに使用す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、コネクターが統合ブローカーへ管理メッセージを送信するときに使用す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` 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ています。

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。サポートされる他の値をリストに追加するには、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 は ICS) のみ有効です。

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後に、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 ICS) 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 を指定することができ、デフォルトは IDL です。
- RepositoryDirectory プロパティーの値がローカル・ディレクトリーの場合、値に使用できるのは JMS のみです。

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。

デフォルト値は JMS です。

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 Espress (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、(サーバー側の) コネクター・コントローラーと (クライアント側の) コネクターの両方を始動するのは困難な場合があります。ご使用のシステムのプロセス・ヒープ・サイズが 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*>) の下の ¥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、「*WebSphere Business Integration Server Express インストール・ガイド*」(Windows 版、Linux 版、または OS/400 および i5/OS 版) 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 ICS 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、次のようになります。
```
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 ICS の場合のみ有効です。

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。

ListenerConcurrency

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

このプロパティーは、MQ トランスポートを使用するコネクターのみで有効です。DeliveryTransport プロパティーの値は MQ でなければなりません。

デフォルト値は 1 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、オプションで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、通貨規格で使用される記号などの国/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、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) 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、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) 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 により起動される 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、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) 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、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) 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、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) 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・アクション間の間隔ではありません。この論理を次に説明します。

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

PollQuantity

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。

E 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。
- 二度目にポーリングされると、コネクターは 2 番目の 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。
- 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 ICS の場合は、この値を <REMOTE> に設定する必要があります。これは、コネクターが InterChange Server 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、これには任意の有効なディレクトリー名を設定することができます。

RequestQueue

RequestQueue プロパティーは、統合ブローカーがコネクターへビジネス・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 (ICS) の場合、サーバーは要求を送信し、JMS 応答キューの応答メッセージを待ちます。

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 ICS 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・フレームワーク用に、JMS ソース・キューを指定します。詳細については、92 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デリバリーします。このキューは、コネクターが同期実行を使用する場合にのみ必要です。同期実行の場合、コネクター・フレームワークは、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。

デフォルト値は *<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、トランスポートでのメッセージ・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. Connector Configurator Express

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

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
- 構成ファイルを作成する
- 構成ファイルのプロパティーを設定する

この付録では、次のトピックについて説明します。

Connector Configurator Express の概要

Connector Configurator Express によって、InterChange Server Express 統合ブローカーで使用するアダプターのコネクター・コンポーネントを構成することができます。

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

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成します。
- **コネクター構成ファイル**を作成します。インストールするコネクターごとに 1 つ構成ファイルを作成する必要があります。
- 構成ファイルのプロパティーを設定します。
  場合によっては、コネクター・テンプレートでプロパティーに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、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。

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、標準プロパティーを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、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、107 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。

Linux でのコネクターの実行

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

Connector Configurator Express 内のいくつかのプロパティーはディレクトリー・パスを使用します。このパスは、Windows のディレクトリー・パスの規則がデフォルトになっています。Linux 環境で構成ファイルを使用する場合、これらのパスの Linux の規則に対応するように、ディレクトリー・パスを修正する必要があります。正しいオペレーティング・システム規則が拡張検証に使用されるように、ツールバー・ドロップ・リストでターゲット・オペレーティング・システムを選択します。

Connector Configurator Express の始動

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

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

スタンドアロン・モードでの Configurator の実行

どのブローカーを実行している場合にも、System Manager を実行せずに Connector Configurator Express を実行し、コネクター構成ファイルを編集できます。

これを行うには、以下のステップを実行します。

- 「**スタート**」>「**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。

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 (113 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。

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。プロジェクトまたはプロジェクトが保管されているディレクトリーからコネクター構成ファイルを選択します。
- 「標準のプロパティー」タブをクリックし、この構成ファイルに含まれているプロパティーを確認します。

コネクター固有のプロパティー・テンプレートの作成

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

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

- テンプレートの新規作成については、107 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でこのテンプレートを保管します。既存のテンプレートは ¥ProductDir¥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**」の下の「**名前**」フィールドに、新規テンプレートの名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
   - テンプレートに含まれているコネクター固有のプロパティー定義を調べるには、「**テンプレート名**」表示でそのテンプレートの名前を選択します。そのテンプレートに含まれているプロパティー定義のリストが「**テンプレートのプレビュー**」表示に表示されます。
3. テンプレートを作成するときには、ご使用のコネクターに必要なプロパティー定義に類似したプロパティー定義が含まれている既存のテンプレートを使用できます。ご使用のコネクターで使用するコネクター固有のプロパティーが表示されるテンプレートが見つからない場合は、自分で作成する必要があります。
   - 既存のテンプレートを変更する場合には、「**変更する既存のテンプレートを選択してください: 検索テンプレート**」の下の「**テンプレート名**」テーブルのリストから、テンプレート名を選択します。
   - このテーブルには、現在使用可能なすべてのテンプレートの名前が表示されます。テンプレートを検索することもできます。

一般特性の指定

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

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

「**プロパティー・タイプ**」がストリングの場合、「**プロパティー・サブタイプ**」を選択できます。これは、構成ファイルの保管時に構文検査を提供するオプションの値です。デフォルトはブランク・スペースで、プロパティーのサブタイプが指定されていないことを意味します。

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

値の指定

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

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

新規プロパティー値を作成するには、以下のステップを実行します。

1. 「値」列見出しの左側の正方形を右マウス・ボタンでクリックします。
2. ポップアップ・メニューから「**追加**」を選択して、「プロパティー値」ダイアログ・ボックスを表示します。ダイアログ・ボックスでは、プロパティー・タイプに応じて、値を入力するか、または値と範囲の両方を入力することができます。
3. 新規プロパティー値を入力し、「OK」をクリックします。右側の「**値**」パネルに値が表示されます。

「**値**」パネルには、3 つの列からなるテーブルが表示されます。

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

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

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

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

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

依存関係の設定

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

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

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

   == (等しい)

!= (等しくない)

> (より大)

< (より小)

>= (より大か等しい)

<= (より小か等しい)

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

パス名の設定

パス名の設定の一般的な規則のいくつかを以下に示します。

- Windows および Linux でのファイル名の最大長は 255 文字です。
- Windows では、絶対パス名は [Drive:][Directory]¥filename の形式に従う必要があります。例えば、C:¥WebSphereAdapters¥bin¥Data¥Std¥StdConnProps.xml のようにします。
  Linux では、最初の文字は / でなければなりません。
- キュー名では、先頭または途中にスペースを使用することはできません。

新しい構成ファイルを作成

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。

ファイルの拡張検証のために、オペレーティング・システムも選択します。ツールバーには「**ターゲット・システム**」というドロップ・リストがあり、ここで、プロパティーの拡張検証用のターゲット・オペレーティング・システムを選択できます。選択可能なオプションは、「Windows」、「Linux」、および「i5/OS」、「その他」 (Windows でも Linux でもない場合)、および「なし (拡張検証なし)」 (拡張検証をオフに切り替え) です。始動時のデフォルトは「Windows」です。

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

- 「System Manager」ウィンドウで、「**ツール**」メニューから「**Connector Configurator Express**」を選択します。Connector Configurator Express が開きます。
- スタンドアロン・モードで、Connector Configurator Express を起動します。

構成ファイルの拡張検証用のオペレーティング・システムを設定するには、以下のステップを実行します。

- メニュー・バーの「**ターゲット・システム:**」ドロップ・リストをプルダウンします。

- 使用中のオペレーティング・システムを選択します。

次に、「**ファイル**」>「**新規**」>「**コネクター構成**」を選択します。「新規コネクター」ウィンドウで、新規コネクターの名前を入力します。

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

- 「**Integration Broker**」フィールドで、ICS 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. メニュー・バーの「**ターゲット・システム:**」ドロップ・リストを使用して、構成ファイルの拡張検証用のオペレーティング・システムを設定します (前述の『新規構成ファイルの作成』を参照してください)。
2. 「**ファイル**」>「**新規**」>「**コネクター構成**」をクリックします。
3. 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と一貫性をもつ一意の名前である必要があります。

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

     「ICS」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。
4. 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。
5. ファイルを保管するには、「**ファイル**」>「**保管**」>「**ファイルに**」をクリックするか、「**ファイル**」>「**保管**」>「**プロジェクトに**」をクリックします。プロジェクトに保管するには、System Manager が実行中である必要があります。ファイ

ルとして保管する場合は、「**ファイル・コネクターを保管**」ダイアログ・ボックスが表示されます。*.cfg をファイル・タイプとして選択し、「ファイル名」フィールド内に名前が正しいスペル (大文字と小文字の区別を含む) で表示されていることを確認してから、ファイルを保管するディレクトリーにナビゲートし、「**保管**」をクリックします。Connector Configurator Express のメッセージ・パネルの状況表示に、構成ファイルが正常に作成されたことが示されます。

重要: ここで設定するディレクトリー・パスおよび名前は、コネクターの始動ファイルで指定するコネクター構成ファイルのパスおよび名前に一致している必要があります。

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

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

既存ファイルを使用してコネクターを構成するには、Connector Configurator 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)
   - ICS リポジトリー (*.in、*.out)

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

- すべてのファイル (*.*)

  コネクターのアダプター・パッケージに *.txt ファイルが付属していた場合、または別の拡張子で定義ファイルが使用可能である場合は、このオプションを選択します。

3. ディレクトリー表示内で、適切なコネクター定義ファイルへ移動し、ファイルを選択し、「**開く**」をクリックします。

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

1. System Manager を始動します。System Manager が開始されている場合にのみ、構成を System Manager から開いたり、System Manager に保管したりできます。
2. Connector Configurator 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。正しいブローカーが設定されていることを確認してください。正しいブローカーが設定されていない場合、コネクターを構成する前にブローカー値を変更してください。これを行うには、以下のステップを実行します。

1. 「**標準のプロパティー**」タブで、BrokerType プロパティーの値フィールドを選択します。ドロップダウン・メニューで、値 ICS を選択します。
2. 選択したブローカーに関連付けられているコネクター・プロパティーが「標準のプロパティー」タブに表示されます。表に、「**プロパティー名**」、「**値**」、「**タイプ**」、「**サブタイプ**」 (「タイプ」がストリングである場合)、「**説明**」、および 「**更新メソッド**」が表示されます。
3. ここでファイルを保管するか、または 117 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。
4. 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。

構成ファイルを作成する前に、プロパティーの拡張検証用のターゲット・オペレーティング・システムを選択することができる「**ターゲット・システム**」ドロップ・リストを使用します。

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。

「**ターゲット・システム**」ドロップ・リストから「Windows」、「Linux」、および「i5/OS」、または「その他」を選択することによって拡張検証機能を使用する場合、システムはタイプだけでなくプロパティー・サブタイプを検証し、検証に失敗した場合は警告メッセージを表示します。

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

- 標準プロパティー
- 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
- セキュリティー

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すべてではなく一部のみです。

- アプリケーション固有のプロパティーは、コネクターのアプリケーション固有コンポーネント (アプリケーションと直接対話するコンポーネント) のみに適用されます。各コネクターには、そのコネクターのアプリケーションだけで使用されるアプリケーション固有のプロパティーがあります。これらのプロパティーには、デフォルト値が用意されているものもあれば、そうでないものもあります。また、一部のデフォルト値は変更することができます。各アダプター・ガイドのインストールおよび構成の章に、アプリケーション固有のプロパティーおよび推奨値が記述されています。

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

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

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下のステップを実行します。

1. 値を設定するフィールド内でクリックします。
2. 値を入力するか、ドロップダウン・メニューが表示された場合にはメニューから値を選択します。

   注: プロパティーの「タイプ」が「ストリング」である場合、「サブタイプ」列にサブタイプ値が含まれている場合があります。このサブタイプは、プロパティーの拡張検証に使用されます。
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。

特定の標準プロパティーに関する詳細を参照するには、「標準のプロパティー」タブ付きシート内のそのプロパティーの「説明」列内の項目を左マウス・ボタンでクリックします。全般ヘルプをインストール済みの場合は、右側に矢印ボタンが表示されます。ボタンをクリックすると、「ヘルプ」ウィンドウが開き、標準プロパティーの詳細が表示されます。

注: ホット・ボタンが表示されない場合、そのプロパティーについては全般ヘルプが見つかっていません。

インストール済みの場合、全般ヘルプ・ファイルは <*ProductDir*>¥bin¥Data¥Std¥Help¥<*RegionalSetting*>¥ にあります。

コネクター固有の構成プロパティーの設定

コネクター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。「**追加**」をクリックしてプロパティーを追加します。子プロパティーを追加するには、親行番号を右マウス・ボタン・クリックして、「**子を追加**」をクリックします。
2. プロパティーまたは子プロパティーの値を入力します。

   注: プロパティーの「タイプ」が「ストリング」である場合、「サブタイプ」ドロップ・リストからサブタイプを選択できます。このサブタイプは、プロパティーの拡張検証に使用されます。
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. 特定のプロパティーに関する詳細を参照するには、そのプロパティーの「説明」列内の項目を左マウス・ボタンでクリックします。全般ヘルプをインストール済みの場合は、ホット・ボタンが表示されます。ホット・ボタンをクリックすると、「ヘルプ」ウィンドウが開き、標準プロパティーの詳細が表示されます。

   注: ホット・ボタンが表示されない場合、そのプロパティーについては全般ヘルプが見つかっていません。
5. 115 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。

全般ヘルプ・ファイルがインストール済みで、AdapterHelpName プロパティーがブランクである場合、Connector Configurator Express は、<*ProductDir*>¥bin¥Data¥App¥Help¥<*RegionalSetting*>¥ にあるアダプター固有の全般ヘルプ・ファイルを指します。それ以外の場合、Connector Configurator Express は、<*ProductDir*>¥bin¥Data¥App¥Help¥<*AdapterHelpName*>¥<*RegionalSetting*>¥にあるアダプター固有の全般ヘルプ・ファイルを指します。標準プロパティーについての付録で説明されている AdapterHelpName プロパティーを参照してください。

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

標準プロパティーについての付録である 83 ページの『標準コネクター・プロパティーの概要』内の『標準コネクター・プロパティーの概要』の下の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 の場合

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

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

1. 「**ビジネス・オブジェクト名**」リストで空のフィールドをクリックします。System Manager プロジェクトに存在するすべてのビジネス・オブジェクト定義を示すドロップ・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 (Integration Component Library) プロジェクトに保管されます。

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

1. ビジネス・オブジェクト・フィールドを選択するため、そのビジネス・オブジェクトの左側の番号をクリックします。
2. 「Connector Configurator 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。

関連付けられたマップ

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

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

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

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

「**関連付けられたマップ**」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的バインディング**

  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 つのマップを見つけて、バインドしようとします。

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

以下のをステップを実行して、マップを明示的にバインドします。

1. 「**明示的 (Explicit)**」列で、バインドするマップのチェック・ボックスにチェックマークを付けます。
2. ビジネス・オブジェクトに関連付けるマップを選択します。
3. 「Connector Configurator 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。

セキュリティー

Connector Configurator Express 内の「**セキュリティー**」タブを使用して、メッセージにさまざまなプライバシー・レベルを設定することができます。DeliveryTransport プロパティーが JMS に設定されている場合のみ、この機能を使用できます。

デフォルトでは、「プライバシー」はオフになっています。使用可能にするには、「**プライバシー**」ボックスにチェック・マークを付けます。

「**鍵ストア・ターゲット・システムの絶対パス名**」は、以下の値です。

- Windows の場合:
  `<ProductDir>¥connectors¥security¥<connectorname>.jks`
- Linux および i5/OS の場合:
  `/ProductDir/connectors/security/<connectorname>.jks`

このパスおよびファイルは、コネクターを始動するシステム、すなわちターゲット・システム上に存在していなければなりません。

ターゲット・システムが現在実行中のシステムである場合のみ、右側の「参照」ボタンを使用できます。「**プライバシー**」が使用可能であり、メニュー・バーの「**ターゲット・システム**」が Windows に設定されている場合を除き、これはグレーアウトされています。

「**メッセージのプライバシー・レベル**」は、3 つのメッセージ・カテゴリー (全メッセージ、全管理メッセージ、および全ビジネス・オブジェクト・メッセージ) で以下のように設定されます。

- “”: がデフォルトです。メッセージ・カテゴリーにプライバシー・レベルが設定されていない場合に使用します。
- none。デフォルトと同じではありません。メッセージ・カテゴリーにプライバシー・レベルなしと故意に設定する場合にこれを使用します。
- integrity
- privacy
- integrity_plus_privacy

「**鍵の保守**」機能によって、サーバーおよびアダプターの公開鍵を生成、インポート、およびエクスポートすることができます。

- 「**鍵の生成**」を選択すると、鍵を生成する keytool のデフォルトを含む「鍵の生成」ダイアログ・ボックスが表示されます。
- 「セキュリティー」タブの「**鍵ストア・ターゲット・システムの絶対パス名**」で入力した値が、鍵ストア値のデフォルトになります。
- 「OK」を選択すると、記入項目が検証され、鍵証明書が生成され、「Connector Configurator Express」ログ・ウィンドウに出力が送られます。

証明書をアダプター鍵ストアにインポートする前に、サーバー鍵ストアからエクスポートする必要があります。「**アダプター公開鍵のエクスポート**」を選択すると、「アダプター公開鍵のエクスポート」ダイアログ・ボックスが表示されます。

- エクスポート証明書のデフォルトは、ファイル拡張子が <filename>.cer であることを除き、鍵ストアと同じ値です。

「**サーバー公開鍵のインポート**」を選択すると、「サーバー公開鍵のインポート」ダイアログ・ボックスが表示されます。

- インポート証明書のデフォルトは、<*ProductDir*>¥bin¥ics.cer になります (システムにファイルが存在する場合)。
- インポート証明書関連はサーバー名でなければなりません。サーバーが登録されていれば、ドロップ・リストからそれを選択することができます。

DeliveryTransport の値が IDL の場合のみ、「**アダプター・アクセス制御**」機能が使用可能です。デフォルトでは、アダプターはゲスト ID を使用してログインします。「**ゲスト ID の使用**」ボックスにチェック・マークが付けられていない場合は、「**アダプター ID**」および「**アダプター・パスワード**」フィールドが使用可能です。

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、そのファイルのログおよびトレースの値をデフォルト値として使用します。これらの値は、Connector Configurator Express 内で変更できます。

ログとトレースの値を変更するには、以下のステップを実行します。

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

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

管」をクリックします。ログ・メッセージまたはトレース・メッセージは、指定した場所の指定したファイルに書き込まれます。

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

データ・ハンドラー

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

これらのプロパティーに使用する値については、付録 A『コネクターの標準構成プロパティー』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、InterChange Server Express が現在使用しているブローカー・モード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、*.con 拡張子付きファイルとして統合コンポーネント・ライブラリーに保管します。
- 指定したディレクトリーに保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに *.cfg 拡張子付きファイルとして保管します。デフォルトでは、このファイルは ¥WebSphereAdapters¥bin¥Data¥App に保管されます。

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、「*システム・インプリメンテーション・ガイド*」を参照してください。

構成ファイルの変更

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

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

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

- Connector Configurator Express で既存の構成ファイルを開きます。

- 「**標準のプロパティー**」タブを選択します。
- 「標準のプロパティー」タブの「**BrokerType**」フィールドで、ご使用のブローカーに合った値を選択します。現行値を変更すると、プロパティー・ウィンドウ内の利用可能なタブおよびフィールド選択がただちに変更され、選択した新規ブローカーに適したタブとフィールドのみが表示されます。

構成の完了

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

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

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

Connector Configurator Express はグローバル化され、構成ファイルと統合ブローカー間の文字変換を処理できます。Connector Configurator 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>
```

付録 C. Adapter for HTTP のチュートリアル

この付録では、以下の手順を段階的に説明します。

- 要求処理とイベント処理の両方の場合の非同期および同期イベント伝送
- HTTP サンプル用に HTTP アダプターを構成する方法
- HTTPS サンプル用に HTTP アダプターを構成する方法

チュートリアルの概要

このチュートリアルは、サポートされた HTTP および HTTPS の両方を持つ Adapter for HTTP で要求処理およびイベント処理の両方を行う場合の、同期および非同期のイベント伝送について説明します。それぞれのシナリオにおけるアダプターの役割は以下のとおりです。

- 外部 URL を呼び出す HTTP クライアント
- URL で HTTP 要求を listen し、それらを InterChange Server Express コラボレーションに送るプロキシー

このチュートリアルでは、以下のサンプル・シナリオを使用してアダプターの基本的な機能を示します。

- **非同期シナリオ** このシナリオでは、非同期 (要求専用) HTTP POST について説明します。このシナリオには 2 つのサンプルがあります。構成を単純にするために、同じ HTTP アダプターを使用して、HTTP 要求を listen し、HTTP クライアントとして URL を呼び出します。
  - **URL で HTTP 要求を listen するプロキシー** このサンプルでは、着信要求は、InterChange Server Express 内のコラボレーションである SERVICE_ASYNCH_Order_Collab に送られます。このコラボレーションは Asynch Order と呼ばれます。アダプターが適切に構成されている場合、HTTP または HTTPS のいずれかのプロトコルを使用して、このコラボレーションを呼び出すことができます。SERVICE_ASYNCH_Order_Collab は SERVICE_ASYNCH_TLO_Order を取る単純パススルー・コラボレーションです。このコラボレーションのトリガー・ポート (元) は HTTPConnector にバインドされています。サービス・ポート (先) は SampleSiebelConnector にバインドされています。
  - **外部 URL を呼び出す HTTP クライアント** このサンプルでは、HTTP クライアントは、InterChange Server Express 内の別のコラボレーション CLIENT_ASYNCH_Order_Collab であり、HTTP アダプターを使用して外部 URL を非同期に呼び出します。アダプターが適切に構成されている場合、この HTTP クライアントは、HTTP または HTTPS のいずれかのプロトコルを使用

して、外部 URL を呼び出すことができます。CLIENT_ASYNCH_Order_Collab は CLIENT_ASYNCH_TLO_Order を取る単純パススルー・コラボレーションです。このコラボレーションのトリガー・ポート (元) は SampleSAPConnector にバインドされています。サービス・ポート (先) は HTTPConnector にバインドされています。

非同期シナリオのいずれのサンプルにも以下の 2 つのアプリケーションが含まれています。

- SampleSiebel: クライアントのためにオーダーを作成します。
- SampleSAP: オーダーを作成します。

- **同期シナリオ** このシナリオでは、同期 (要求/応答) HTTP POST について説明します。このシナリオには 2 つのサンプルがあります。構成を単純にするために、同じ HTTP アダプターを使用して、HTTP 要求を listen し、HTTP クライアントとして URL を呼び出します。
  - **URL で HTTP 要求を listen するプロキシー** このサンプルでは、着信要求は、InterChange Server Express 内のコラボレーションである SERVICE_SYNCH_OrderStatus_Collab に送られます。このコラボレーションは Synch OrderStatus と呼ばれます。アダプターが適切に構成されている場合、HTTP または HTTPS のいずれかのプロトコルを使用して、このコラボレーションを呼び出すことができます。SERVICE_SYNCH_OrderStatus_Collab は SERVICE_SYNCH_TLO_OrderStatus を取る単純パススルー・コラボレーションです。このコラボレーションのトリガー・ポート (元) は HTTPConnector にバインドされています。サービス・ポート (先) は SampleSiebelConnector にバインドされています。
  - **外部 URL を呼び出す HTTP クライアント** このサンプルでは、HTTP クライアントは、InterChange Server Express 内の別のコラボレーション CLIENT_SYNCH_OrderStatus_Collab であり、HTTP アダプターを使用して外部 URL を呼び出します。アダプターが適切に構成されている場合、この HTTP クライアントは、HTTP または HTTPS のいずれかのプロトコルを使用して、外部 URL を呼び出すことができます。CLIENT_SYNCH_OrderStatus_Collab は CLIENT_SYNCH_TLO_OrderStatus を取る単純パススルー・コラボレーションです。このコラボレーションのトリガー・ポート (元) は SampleSAPConnector にバインドされています。サービス・ポート (先) は HTTPConnector にバインドされています。

  同期シナリオのいずれのサンプルにも以下の 2 つのアプリケーションが含まれています。

  - SampleSiebel: クライアントのためにオーダーの状況を検索します。
  - SampleSAP: オーダーの状況を要求します。

いずれのシナリオでも、2 つの Test Connector を使用して SampleSiebelConnector および SampleSAPConnector をシミュレートします。

はじめに

チュートリアルを開始する前に、以下のことを確認してください。

- InterChange Server Express 4.2.2 以降がインストールされ、その運用経験を持っていること。

- InterChange Server Express ホーム・ディレクトリーに WebSphere Business Integration Server Express Adapter for HTTP をインストール済みであること。
- HTTP テクノロジーの運用経験があること。
- XML テクノロジーの運用経験があること。

インストールと構成

以下のセクションでは、*WBI_folder* は現在インストールされている InterChange Server Express が格納されているフォルダーを指します。すべての環境変数およびファイル分離文字は Windows 2003 の形式で記述されます。

サーバーとツールの始動

1. ショートカットから InterChange Server Express を始動します。
2. WebSphere Business Integration Server Express System Manager を始動して、Component Navigator のパースペクティブを開きます。
3. サーバーをサーバー・インスタンスとして InterChange Server Express ビューに登録および接続します。

サンプル・コンテンツのロード

Component Navigator のパースペクティブから以下を実行します。

1. 新規統合コンポーネント・ライブラリーの作成
2. WBI_folder¥connectors¥HTTP¥samples¥WebSphereICS¥ にある HTTPSample.jar というリポジトリー・ファイルのインポート

コラボレーション・テンプレートのコンパイル

WebSphere Business Integration Server Express System Manager を使用して以下の作業を実行します。

- リポジトリー・ファイル HTTPSample.jar からインポートされたコラボレーション・テンプレートを**すべてコンパイル**する。

コネクターの構成

1. コネクターをまだ構成していない場合は、このガイドの説明に従ってご使用のシステムに応じて構成してください。
2. WebSphere Business Integration Server Express System Manager を使用して、Connector Configurator Express の HTTPConnector を開きます。
3. また、このサンプルで使用するプロトコルの HTTPConnector も構成する必要があります。
   - HTTP を使用する場合は、128 ページの『HTTP プロトコル・シナリオ用の構成』を参照して HTTP 用にコネクターを構成してください。
   - HTTPS を使用する場合は、128 ページの『HTTPS プロトコル・シナリオ用の構成』を参照して HTTPS 用にコネクターを構成してください。

HTTP プロトコル・シナリオ用の構成

このセクションでは、HTTP サンプル・シナリオ用にコネクターを構成する方法について説明します。本書の本文に記載されているとおり、コネクターには HTTP プロトコル・リスナーと HTTP-HTTPS プロトコル・ハンドラーが組み込まれています。

以下のステップと説明では、階層コネクター構成プロパティーが -> 記号で示されています。例えば、A-> B は、A が階層プロパティーであり、B が A の子プロパティーであることを示します。

このサンプルに対して HTTP プロトコル・リスナーを構成するには、以下の手順に従います。

1. Connector Configurator Express で、HTTPConnector の「**コネクター固有プロパティー**」をクリックします。
2. **ProtocolListenerFramework** プロパティーを展開して、ProtocolListeners 子プロパティーを表示します。
3. **ProtocolListeners** 子プロパティーを展開して、**HTTPListener1** 子プロパティーを表示します。
4. **HTTPListener1->Host** および **HTTPListener1->Port** の各プロパティーの値を検査します。ホストで他のプロセスが実行中でないこと、この TCP/IP ポートで listen していないことを確認してください。オプションで、**HTTPListener1->Host** の値をコネクターを実行するマシン名に設定することもできます。

このためにこのサンプルに HTTP-HTTPS プロトコル・ハンドラーを構成する必要はありません。

HTTPS プロトコル・シナリオ用の構成

このセクションでは、HTTPS サンプル・シナリオ用にコネクターを構成する方法について説明します。コネクターには、HTTPS プロトコル・リスナーおよび HTTP-HTTPS プロトコル・ハンドラーが含まれています。

以下のステップと説明では、階層コネクター構成プロパティーが -> 記号で示されています。例えば、A-> B は、A が階層プロパティーであり、B が A の子プロパティーであることを示します。

注: 126 ページの『はじめに』にリストされたプリインストール項目の他に、鍵および証明書管理ソフトウェアを使用して、鍵ストアとトラストストアを作成およびテストしておく必要があります。

SSL コネクター固有プロパティーの構成: HTTPS の場合、コネクターに SSL コネクター固有の階層プロパティーを構成する必要があります。

1. Connector Configurator Express で、HTTPConnector の「**コネクター固有プロパティー**」タブをクリックします。
2. **SSL** 階層プロパティーを展開して、すべての子プロパティーを表示します。さらに、階層 SSL コネクター固有プロパティーの以下の子プロパティーを確認または変更します。

- **SSL->KeyStore** このプロパティーは、鍵ストア・ファイルへの完全パスに設定します。このファイルは鍵および証明書管理ソフトウェアを使用して作成する必要があります。
- **SSL->KeyStorePassword** このプロパティーは、KeyStore にアクセスするために必要なパスワードに設定します。
- **SSL->KeyStoreAlias** このプロパティーは、KeyStore の秘密鍵の別名に設定します。
- **SSL->TrustStore** このプロパティーは、鍵および証明書管理ソフトウェアを使用して作成したトラストストア・ファイルへの完全パスに設定します。
- **SSL->TrustStorePassword** このプロパティーは、TrustStore にアクセスするために必要なパスワードに設定します。

注: 必ず Connector Configurator Express で変更内容を保管するようにしてください。

HTTPS プロトコル・リスナーの構成:

1. Connector Configurator Express で、HTTPConnector の「**コネクター固有プロパティー**」をクリックします。
2. **ProtocolListenerFramework** プロパティーを展開して、**ProtocolListeners** 子プロパティーを表示します。
3. **ProtocolListeners** 子プロパティーを展開して、**HTTPSListener1** 子プロパティーを表示します。**HTTPSListener1->Host** および **HTTPSListener1->Port** の各プロパティーの値を検査します。ホストで他のプロセスが実行中でないこと、この TCP/IP ポートで listen していないことを確認してください。オプションで、**HTTPSListener1->Host** の値をコネクターを実行するマシン名に設定することもできます。

このためにこのサンプルに HTTP-HTTPS プロトコル・ハンドラーを構成する必要はありません。

KeyStore および TrustStore のセットアップ: サンプル・シナリオで使用できるように KeyStore と TrustStore を素早くセットアップできます。実動システムの場合は、鍵ストア、証明書、および鍵生成をセットアップおよび管理するために、サード・パーティーのソフトウェアを使用する必要があります。これらのリソースをセットアップおよび管理するためのツールは Adapter for HTTP には含まれていません。

このセクションは、Java 仮想マシンがシステムにインストール済みであること、JVM (Java 仮想マシン) に付属の鍵ツールについて十分な知識と経験があることを前提としています。鍵ツールの詳細と問題のトラブルシューティングについては、JVM に付属の資料を参照してください。

KeyStore をセットアップするには、以下の手順に従います。

1. 鍵ツールを使用して KeyStore を作成します。KeyStore に鍵ペアを作成する必要があります。これを実行するには、コマンド行に以下を入力します。

   ```
   keytool -genkey -alias httpadapter -keystore c:¥security¥keystore
   ```

2. これにより、鍵ツールからパスワードを入力するようプロンプトが出ます。SSL->KeyStorePassword コネクター・プロパティーの値として入力したパスワー

ドを指定します。
上記の例で、コマンド行に -keystore c:¥security¥keystore を指定した場合は、c:¥security¥keystore を SSL->KeyStore プロパティーの値として入力することができます。また、コマンド行に -alias httpadapter を指定した場合は、httpadapter を SSL->KeyStoreAlias コネクター・プロパティーの値として入力することができます。次に、鍵ツールから、証明書の詳細についてプロンプトが出されます。以下は、各プロンプトとその入力例を示しています。ただし、以下に示すのは単なる例にすぎません。必ず鍵ツールの資料を参照してこれに従ってください。

```
What is your first and last name?
     [Unknown]:  HostName
What is the name of your organizational unit?
     [Unknown]:  myunit
What is the name of your organization?
     [Unknown]:  myorganization
What is the name of your City or Locality?
     [Unknown]:  mycity
What is the name of your State or Province?
     [Unknown]:  mystate
What is the two-letter country code for this unit?
     [Unknown]:  mycountryIs <CN=HostName, OU=myunit, O=myorganization,
     L=mycity, ST=mystate, C=mycountry> correct?
     [no]:  yes
```

3. 「What is your first and last name?」というプロンプトが表示されたら、コネクターを実行しているマシンの名前を入力します。次に、鍵ツールから次のプロンプトが出されます。

   ```
   Enter key password for <httpadapter> (RETURN if same as keystore password):
   ```

4. 同じパスワードを使用するには「**Return**」を選択します。自己署名証明書を使用する場合は、上記で作成済みの証明書のエクスポートが可能です。これを実行するには、コマンド行に以下を入力します。

   ```
   C:¥security>keytool -export -alias httpadapter -keystore c:¥security¥keystore
   -file c:¥security¥httpadapter.cer
   ```

5. ここで、鍵ツールから、鍵ストア・パスワードを入力するよう求めるプロンプトが出されます。上記で入力したパスワードを入力します。

TrustStore をセットアップするには、以下の手順に従います。

1. TrustStore にトラステッド証明書をインポートするには、次のコマンドを入力します。

   ```
   keytool -import -alias trusted1 -keystore c:¥security¥truststore
   -file c:¥security¥httpadapter.cer
   ```

2. ここで、鍵ツールから、鍵ストア・パスワードを入力するよう求めるプロンプトが出されます。-keystore c:¥security¥truststore と入力した場合は、SSL->TrustStore プロパティーが c:¥security¥truststore に設定されていることを必ず確認してください。また、SSL->TrustStorePassword プロパティーの値を、上記で入力したパスワードに設定する必要があります。

ユーザー・プロジェクトの作成

- WebSphere Business Integration System Manager を使用して、新規**ユーザー・プロジェクト**を作成します。 127 ページの『サンプル・コンテンツのロード』で作成した統合コンポーネント・ライブラリーから、すべてのコンポーネントを選択します。

プロジェクトの追加と配置

1. サーバー・インスタンス・ビューから、 130 ページの『ユーザー・プロジェクトの作成』で作成した**ユーザー・プロジェクト**を InterChange Server Express に追加します。
2. このユーザー・プロジェクトのすべてのコンポーネントを InterChange Server Express に配置します。

InterChange Server Express のリブート

1. すべての変更内容を有効にするため、InterChange Server Express をリブートします。
2. System Monitor ツールを使用して、コラボレーション・オブジェクト、コネクター・コントローラー、およびマップがすべて正常であることを確認します。

非同期シナリオの実行

このシナリオでは、Asynch Order Service HTTP サービスを呼び出します。シナリオを実行する前に、シナリオに関するデータの流れを段階を追って説明します。

1. CLIENT_ASYNCH_TLO_Order.Create イベントが、テスト・コネクターの 1 つのインスタンスで動作する SampleSAP という名前のアプリケーションで発生します。
2. イベントが SampleSAP から CLIENT_ASYNCH_Order_Collab という名前のコラボレーションに送信されます。
3. 続いて、イベントがコラボレーションから HTTPConnector に送信されます。
4. HTTPconnector は次に、CLIENT_ASYNCH_TLO_Order オブジェクトの子である XML_Order オブジェクトを見つけます。
5. 要求ビジネス・オブジェクトが、XML データ・ハンドラーを使用して XML メッセージに変換されます。 HTTPconnector は、プロトコル構成メタオブジェクト (MO) の Destination 属性によって提供された URL に XML メッセージを送信します。コネクターが使用するプロトコル構成 MO は、CLIENT_ASYNCH_TLO_Order の Handler 属性の値によって決まります。この値は http または https に設定されます。
6. XML 要求は URL にPOST されます。前に説明したように、同じ HTTPConnector が、同じ URL で XML 要求を listen します。コネクターのプロトコル・リスナーが XML メッセージを受け取ります。
7. コネクターは XML メッセージを XML_Order に変換してから、SERVICE_ASYNCH_TLO_Order オブジェクトを作成します。XML_Order オブジェクトが SERVICE__ASYNCH_TLO_Order オブジェクトの子として設定されます。
8. HTTPConnector は、SERVICE_TLO_Order オブジェクトを非同期で InterChange Server Express に通知するようになりました。これにより、非同期 URL の呼び出しが完了します。

これは非同期呼び出し (要求専用) であるため、HTTP クライアントに応答は戻されません。SERVICE_ASYNCH_Order_Collab がこのオブジェクトを受信すると、コラボレーションがビジネス・オブジェクトを Test Connector の 2 番目のインスタンスとして実行中の SampleSiebel というアプリケーションに送信します。オブジェク

トが Test Connector に表示されます。SampleSiebel アプリケーションから「正常に応答」を選択すると、イベントが SERVICE_ASYNCH_Order_Collab に戻されます。

非同期シナリオを実行するには、以下の手順を実行します。

1. InterChange Server Express 統合ブローカーがまだ稼働していない場合は始動します。
2. HTTP コネクターを始動します。
3. Test Connector の 2 つのインスタンスを開始します。
4. Test Connector を使用して、SampleSAPConnector および SampleSiebelConnector のプロファイルを定義します。
5. エージェントのシミュレートを開始するため、それぞれの「Test Connector」メニューから「**ファイル**」->「**エージェントの接続**」を選択します。
6. Test Connector を使用して SampleSAPConnector をシミュレートしながら、メニューから「**編集**」->「**ビジネス・オブジェクトのロード**」を選択します。以下のファイルをロードします。

   WBI_folder¥connectors¥HTTP¥samples¥WebSphereICS¥OrderStatus¥CLIENT_ASYNCH_TLO_Order.bo

   Test Connector は CLIENT_ASYNCH_TLO_Order がロードされたというメッセージを表示します。
7. HTTP URL アドレスを確認します。
   - HTTP サンプルを実行するには、以下の手順を実行します。
     a. Test Connector で、CLIENT_ASYNCH_TLO_Order ビジネス・オブジェクトの Handler 属性の値が http に設定されていることを確認します。
     b. CLIENT_ASYNCH_TLO_Order の Request 属性を展開します。この属性のタイプは CLIENT_ASYNCH_Order ビジネス・オブジェクトです。
     c. XML_Order の HTTPCfgMO 属性を展開します。この属性のタイプは XML_Order_HTTP_CfgMO です。
     d. XML_Order_HTTP_CfgMO の Destination 属性の値が http://localhost:8080/wbia/http/samples に設定されていることを確認します。
   - **HTTPS サンプルを実行するには、以下の手順を実行します。**
     a. HTTPS 呼び出しであっても、CLIENT_ASYNCH_TLO_Order ビジネス・オブジェクトの Handler 属性の値が http に設定されていることを確認します。
     b. CLIENT_ASYNCH_TLO_Order の Request 属性を展開します。この属性のタイプは XML_Order ビジネス・オブジェクトです。
     c. XML_Order の HTTPCfgMO 属性を展開します。この属性のタイプは XML_Order_HTTP_CfgMO です。
     d. XML_Order_HTTP_CfgMO の Destination 属性の値が https://localhost:443/wbia/http/samples に設定されていることを確認します。
8. Test Connector を使用して SampleSAPConnector をシミュレートしながら、ロードした**テスト・ビジネス・オブジェクト**をクリックします。メニューから

「**要求**」>「**送信**」を選択します。イベントの流れについての詳細は、このセクションで前述した段階的な説明を参照してください。

9. Test Connector を使用して SampleSiebelConnector をシミュレートしながら、「**要求**」->「**要求の受付**」を選択します。Test Connector の右パネルに、SERVICE_ASYNCH_TLO_Order.Create というラベルの付いたイベントが表示されます。
10. ビジネス・オブジェクトをダブルクリックします。ウィンドウ内でビジネス・オブジェクトが開きます。
11. ビジネス・オブジェクトの Request 属性を展開します。Request 属性のタイプは SERVICE_ASYNCH_Order です。SERVICE_ASYNCH_Order の OrderId、CustomarId およびその他の属性をインスペクションして、受信したオーダーを検査します。これにより、非同期シナリオの実行が完了します。
12. ビジネス・オブジェクトのインスペクションが完了したら、ウィンドウを閉じます。「**要求**」->「**返信**」->「**成功**」を選択します。

同期シナリオの実行

このシナリオでは、Synch OrderStatus Service HTTP サービスを呼び出します。シナリオを実行する前に、シナリオに関するデータの流れを段階を追って説明します。

1. CLIENT_SYNCH_TLO_OrderStatus.Retrieve イベントが、Test Connector の 1 つのインスタンスで動作する SampleSAP という名前のアプリケーションで発生します。
2. イベントが SampleSAP から CLIENT_SYNCH_OrderStatus_Collab という名前のコラボレーションに送信されます。
3. 続いて、イベントがコラボレーションから HTTP コネクターに送信されます。
4. HTTP コネクターは、CLIENT_SYNCH_TLO_OrderStatus の子要求である XML_OrderStatus オブジェクトを見つけます。
5. HTTP コネクターは、XML データ・ハンドラーを呼び出して XML_OrderStatus ビジネス・オブジェクトを XML メッセージに変換します。
6. XML 要求は URL にPOST されます。前に説明したように、同じ HTTPConnector が、同じ URL で XML 要求を listen します。コネクターのプロトコル・リスナーが XML メッセージを受け取ります。
7. コネクターのプロトコル・リスナーは XML メッセージを XML_OrderStatus に変換してから、SERVICE_SYNCH_TLO_Order オブジェクトを作成します。XML_OrderStatus オブジェクトが SERVICE_SYNCH_TLO_Order オブジェクトの子として設定されます。
8. 次に、HTTP コネクターが SERVICE_SYNCH_TLO_OrderStatus オブジェクトを InterChange Server Express で実行中の SERVICE_SYNCH_OrderStatus_Collab コラボレーションに同期的に通知します。これは同期実行であるため、コラボレーションが実行されて応答が戻るまで、HTTP コネクターはブロックされたままになります。
9. 次に、HTTP コネクターが SERVICE_TLO_OrderStatus オブジェクトを InterChange Server Express で実行中の SERVICE_SYNCH_OrderStatus_Collab コ

ラボレーションに同期的に通知します。これは同期実行であるため、コラボレーションが実行されて応答が戻るまで、HTTP コネクターはブロックされたままになります。

10. 値を編集し、SampleSiebel アプリケーションから Reply Success を選択すると、イベントは SERVICE_SYNCH_OrderStatus_Collab コラボレーションに戻されます。
11. SERVICE_SYNCH_OrderStatus_Collab が SERVICE_SYNCH_TLO_OrderStatus オブジェクトを受信します。次に、コラボレーションがビジネス・オブジェクトを HTTPConnector に送信します。
12. HTTPConnector は、SERVICE_SYNCH_OrderStatus_TLO の子である XML_OrderStatus ビジネス・オブジェクトを見つけます。このビジネス・オブジェクトは、XML データ・ハンドラーによって XML 応答メッセージに変換されます。
13. XML 応答は、HTTP クライアントに送り戻されます。
14. HTTP クライアント (この場合はHTTP コネクターのプロトコル・ハンドラー) が応答を受信します。コネクターが応答メッセージによって XML データ・ハンドラーを呼び出します。 XML データ・ハンドラーは、応答メッセージを XML_OrderStatus ビジネス・オブジェクトに変換します。HTTPConnector がこのオブジェクトを CLIENT_SYNCH_OrderStatus_TLO の子として設定します。
15. CLIENT_SYNCH_OrderStatus_TLO が CLIENT_SYNCH_OrderStatus_Collab コラボレーションに戻されます。
16. 次に、CLIENT_SYNCH_OrderStatus_Collab が CLIENT_SYNCH_OrderStatus_TLO を Test Connector の 1 番目のインスタンスとして実行中の SampleSAP というアプリケーションに送信します。Test Connector がこのオブジェクトを表示します。

同期シナリオを実行するには、以下の手順を実行します。

1. InterChange Server Express 統合ブローカーがまだ稼働していない場合は始動します。
2. HTTP コネクターを始動します。
3. Test Connector の 2 つのインスタンスを開始します。
4. Test Connector を使用して、SampleSAPConnector および SampleSiebelConnector のプロファイルを定義します。
5. エージェントのシミュレートを開始するため、それぞれの「Test Connector」メニューから「**ファイル**」->「**エージェントの接続**」を選択します。
6. Test Connector を使用して SampleSAPConnector をシミュレートしながら、メニューから「**編集**」->「**ビジネス・オブジェクトのロード**」を選択します。以下のファイルをロードします。

   WBI_folder¥connectors¥HTTP¥samples¥WebSphereICS¥OrderStatus¥CLIENT_SYNCH_TLO_OrderStatus.bo

   Test Connector は CLIENT_SYNCH_TLO_OrderStatus がロードされたというメッセージを表示します。
7. HTTP URL アドレスを確認します。

   • **HTTP サンプルを実行するには、以下の手順を実行します。**

a. Test Connector で、CLIENT_SYNCH_TLO_OrderStatus ビジネス・オブジェクトの Handler 属性の値が http に設定されていることを確認します。

b. CLIENT_SYNCH_TLO_OrderStatus の Request 属性を展開します。この属性のタイプは XML_OrderStatus ビジネス・オブジェクトです。

c. XML_OrderStatus の HTTPCfgMO 属性を展開します。この属性のタイプは XML_Order_HTTP_CfgMO です。

d. XML_Order_HTTP_CfgMO の Destination 属性の値が http://localhost:8080/wbia/http/samples に設定されていることを確認します。

- **HTTPS サンプルを実行するには、以下の手順を実行します。**

a. Test Connector で、https 呼び出しであっても、CLIENT_SYNCH_TLO_OrderStatus ビジネス・オブジェクトの Handler 属性の値が http に設定されていることを確認します。

b. CLIENT_SYNCH_TLO_OrderStatus の Request 属性を展開します。この属性のタイプは XML_OrderStatus ビジネス・オブジェクトです。

c. XML_OrderStatus の HTTPCfgMO 属性を展開します。この属性のタイプは XML_Order_HTTP_CfgMO です。

d. XML_Order_HTTP_CfgMO の Destination 属性の値が https://localhost:443/wbia/http/samples に設定されていることを確認します。

8. Test Connector を使用して SampleSAPConnector をシミュレートしながら、ロードした**テスト・ビジネス・オブジェクト**をクリックします。メニューから「**要求**」>「**送信**」を選択します。データの流れについての詳細は、このセクションで前述した段階的な説明を参照してください。
9. SampleSiebelConnector をシミュレートしている Test Connector インスタンスの右パネルに、SERVICE_SYNCH_TLO_OrderStatus.Retrieve というラベルの付いたイベントが表示されます。ビジネス・オブジェクトをダブルクリックしてウィンドウに表示します。
10. ビジネス・オブジェクトの Request 属性を展開します。要求の値を検査し、SampleSAPConnector から送られた値が完全な状態にあることを確認します。
11. **LOAD BO** を選択し、ビジネス・オブジェクトの応答属性にデータを取り込みます。以下のファイルをロードします。

    - WBI_folder¥connectors¥HTTP¥samples¥WebSphereICS¥SERVICE_SYNCH_TLO_OrderStatus.bo

    Test Connector は SERVICE_SYNCH_TLO_OrderStatus がロードされたというメッセージを表示します。
12. 「**要求**」->「**返信**」->「**成功**」を選択します。
13. SampleSAPConnector をシミュレートしている Test Connector の右パネルに SERVICE_SYNCH_TLO_OrderStatus.Retrieve というラベルの付いたイベントが表示されます。
14. **SERVICE_SYNCH_TLO_OrderStatus.Retrieve** ビジネス・オブジェクトをダブルクリックします。ダブルクリックすると、ビジネス・オブジェクトがウィンドウに表示されます。SampleSiebelConnector によってオーダーの状況が戻された

場合は、取り込まれたビジネス・オブジェクトの Response 属性が表示されます。**Response** 属性を展開してオーダーの状況を確認します。

15. ビジネス・オブジェクトのインスペクションが完了したら、ウィンドウを閉じます。「**要求**」->「**返信**」->「**成功**」を選択します。

これにより、同期シナリオの実行が完了します。

付録 D. HTTPS/SSL の構成

- 『鍵ストアのセットアップ』
- 138 ページの『トラストストアのセットアップ』
- 139 ページの『公開鍵証明書用の証明書署名要求 (CSR) の生成』

SSL の使用を計画している場合は、鍵ストア、証明書、および鍵生成を管理するために、サード・パーティーのソフトウェアを使用する必要があります。HTTP コネクターにはこれらの作業用のツールは備わっていません。ただし、IBM JRE に同梱の Keytool の使用を選択して、自己署名証明書を作成し、鍵ストアを管理することもできます。

鍵および証明書管理ユーティリティー、すなわち鍵ツールにより自己所有の公開鍵/秘密鍵の鍵ペアおよび関連証明書の管理が可能です。これらは、自己認証 (他のユーザーまたはサービスに対して自分自身を認証させる)、またはデータ保全性もしくはディジタル署名を使用する認証サービス用に使用されます。さらに鍵ツール・ユーティリティーでは、通信するピアの公開鍵を証明書の形態で保管することが可能です。

この付録では、鍵ツールを使用して鍵ストアのセットアップを行う方法について説明します。この付録は、具体的な例の説明のみを意図しており、鍵ツールまたは関連製品の資料を置き換えるものではありませんので、ご注意ください。鍵ストアのセットアップでツールを使用する際は、必ずソース資料を参照してください。鍵ツールの詳細については、次の web サイトを参照してください。

- http://java.sun.com/j2se/1.3/docs/tooldocs/tools.html#security

鍵ストアのセットアップ

鍵ツールを使用して鍵ストアを作成するには、まず最初に鍵ストアに鍵ペアを作成する必要があります。例えば、コマンド行に次のように入力します。

```
keytool -genkey -alias httpadapter -keystore c:¥security¥keystore
```

これにより、鍵ツールからパスワードを入力するよう求めるプロンプトが出されます。鍵ツール・パラメーター内で選択したパスワードを入力することも可能ですが、鍵ツールで入力したパスワードは、SSL ” KeyStorePassword コネクター・プロパティーの値として指定する必要があります。詳しくは、 74 ページの『KeyStorePassword』を参照してください。

このコマンド例は、`c:¥security¥keystore` ディレクトリーで命名された `keystore` の鍵ストアを作成します。したがって、`c:¥security¥keystore` を SSL ” KeyStore コネクター階層プロパティーの値として入力することができます。また上記例のように、コマンド行から `-alias httpadapter` を SSL ” KeyStoreAlias コネクター階層プロパティーの値として入力することもできます。次に、鍵ツール・ユーティリティーから、証明書の詳細についてプロンプトが出されます。以下は、各プロンプトとその入力例を示しています。(鍵ツール文書を参照してください。)

```
What is your first and last name?
     [Unknown]:  HostName
What is the name of your organizational unit?
     [Unknown]:  wbi
What is the name of your organization?
     [Unknown]:  IBM
What is the name of your City or Locality?
     [Unknown]:  Burlingame
What is the name of your State or Province?
     [Unknown]:  CA
What is the two-letter country code for this unit?
    [Unknown]:  US
Is <CN=HostName, OU=wbi, O=IBM, L=Burlingame,
ST=CA, C=US> correct?
    [no]:  yes
```

鍵ツールからパスワードを入力するよう求める次のプロンプトが出されます。

```
Enter key password for <httpadapter> (RETURN if same as keystore password):
```

同じパスワードを使用するには「Return」を選択します。自己署名証明書を使用する場合は、上記で作成済みの証明書のエクスポートが可能です。その場合は、コマンド行で次のように入力します。

```
keytool -export -alias httpadapter -keystore c:¥security¥keystore -file
wsadapter.cer
```

ここで、鍵ツールから、鍵ストア・パスワードを入力するよう求めるプロンプトが出されます。上記で入力したパスワードを入力します。

トラストストアのセットアップ

以下のようにして、TrustStore をセットアップすることができます。

HTTPS プロトコル・リスナーにクライアントを認証させたい場合、SSL の UseClientAuth コネクター構成プロパティーを `true` に設定してください。このケースでは、HTTPS プロトコル・リスナーは、すべてのトラステッド・クライアントに対応した証明書を、トラストストアが収容するよう要求します。コネクターは、JSSE デフォルト・メカニズムを使用してクライアントを信頼することに注意してください。

HTTPS サービスを呼び出す場合、HTTP-HTTPS プロトコル・ハンドラーは、TrustStore がサービスを信頼することを要求します。これは、トラストストアがトラステッド HTTP サービスの証明書をすべて収容していなければならないことを意味しています。 コネクターは、JSSE デフォルト・メカニズムを使用してクライアントを信頼することに注意してください。トラストストアにトラステッド証明書をインポートするには、次のようにコマンドを入力します。

```
keytool -import -alias trusted1 -keystore c:¥security¥truststore -file
 c:¥security¥trusted1.cer
```

ここで、鍵ツールから、鍵ストア・パスワードを入力するよう求めるプロンプトが出されます。`-keystore c:¥security¥truststore` と入力する場合は、SSL -> TrustStore 階層プロパティーが `c:¥security¥truststore` に設定されていることを必ず確認してください。また、SSL -> TrustStorePassword 階層プロパティーの値を、前に入力したパスワードに設定する必要があります。

公開鍵証明書用の証明書署名要求 (CSR) の生成

ユーザーの身元を信頼しているトラステッド・パートナーの中での SSL データ交換では、自己署名証明書は適切と考えられます。ただし、証明書が証明機関 (CA) により署名されていれば、より他者から信頼されやすくなります。

鍵ツール・ユーティリティーを使用して CA で署名された証明書を取得するには、まず最初に Certificate Signing Request (CSR) を生成し、その CSR を CA に渡します。次に、CA は証明書に署名してユーザーに戻します。

次のコマンドの入力により CSR が生成されます。

```
keytool -certreq -alias wsadapter -file httpadapter.csr
 -keystore c:¥security¥keystore
```

このコマンドで、alias は、秘密鍵用に作成した鍵ストアの別名です。鍵ツール・ユーティリティーは CA に提供する CSR ファイルを生成します。次に CA より署名済みの証明書が提供されます。この証明書を鍵ストアにインポートする必要があります。そのためには、次のコマンドを入力します。

```
keytool -import -alias wsadapter -keystore c:¥security¥keystore -trustcacerts
-file casignedcertificate.cer
```

一度インポートすると、鍵ストアの自己署名証明書は CA 署名済み証明書で置き換えられます。

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。汎用プログラミング・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
IMS
i5/OS
Informix
iSeries
Lotus
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

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