IBM WebSphere Business Integration
Adapters

Adapter for Trading Partner Interchange
ユーザーズ・ガイド

バージョン 3.4.x

IBM WebSphere Business Integration
Adapters

Adapter for Trading Partner Interchange
ユーザーズ・ガイド

バージョン 3.4.x

お願い

本書および本書で紹介する製品をご使用になる前に、 89 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、コネクター・バージョン 3.4.x、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： IBM WebSphere Business Integration Adapters
Adapter for Trading Partner Interchange
User Guide
Version 3.4.x

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第2刷 2004.3

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

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

目次

本書について

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

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

対象読者

この資料は、WebSphere Business Integration システムの一部としてコネクターを実装する WebSphere のコンサルタントやお客様を対象にしています。この資料に記載されている情報を利用するには、以下の分野に関する十分な知識が必要です。

- コネクター開発
- ビジネス・オブジェクト開発

関連文書

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

以下のサイトから、関連資料をインストールすることができます。

アダプターの一般情報が必要な場合、アダプターを WebSphere Message Broker (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) とともに使用する場合、およびアダプターを WebSphere Application Server とともに使用する場合は、以下のサイトを参照してください。

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

アダプターを InterChange Server とともに使用する場合は、以下のサイトを参照してください。

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

Message Broker (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) の詳細については、以下のサイトを参照してください。

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

WebSphere Application Server の詳細については、以下を参照してください。

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

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

表記上の規則

この資料は下記の規則に従って編集されています。

`Courier フォント`	コマンド名、ファイル名、入力情報、システムが画面に出力した情報など、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	青のアウトラインは、マニュアルをオンラインで表示するときにのみ見られるもので、相互参照用のハイパーリンクを示します。アウトラインの内側をクリックすることにより、参照先オブジェクトにジャンプできます。
{ }	構文の説明で、複数のオプションが中括弧で囲まれている場合は、その中の 1 つのオプションのみを選択することが必要です。
[ ]	構文の記述行で、大括弧 ([ ]) で囲まれた部分はオプションのパラメーターです。
...	構文の説明で、省略符号は、直前のパラメーターの繰り返しを示します。例えば、`option[,...]` は、複数のオプションをコンマで区切って指定できることを示します。
< >	命名規則により、1 つの名前の個々の要素を互いに区別するために、不等号括弧によって個々の要素が囲まれます。例えば、`<server_name><`*`connector_name`*`>tmp.log` のように使用します。
/、¥	本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。UNIX システムの場合には、円記号をスラッシュ (/) に置き換えてください。すべての WebSphere Business Integration システム製品のパス名は、使用システム上に存在する製品のインストール先ディレクトリーに対する相対パスになっています。
%text% および *$text*	パーセント記号で囲まれたテキストは、Windows の `text` システム変数またはユーザー変数の値を示します。UNIX 環境での同等表記は、*$text* です。*text* の値は、UNIX 環境変数を示します。
ProductDir	製品がインストールされるディレクトリーを表します。

本リリースの新機能

リリース 3.4x の新機能

2004 年 2 月

RosettaNet Implementation Framework (RNIF) over TPI が提供されています。

実装シナリオのサンプルが提供されています。

2003 年 12 月

Adapter for Trading Partner Interchange は一般的な保守フィックスにより更新されました。

バージョン 3.4 以降の Adapter for Trading Partner Interchange は Microsoft Windows NT および IBM AIX ではサポートされなくなりました。

アダプターのインストール情報は、本書から移動しました。この情報の新たな入手先については、8 ページの『TPI Server のインストール』 (第 2 章)を参照してください。

以前のバージョン

以下は、以前のバージョンにおける変更点および新規機能の履歴です。

バージョン 3.3.x

アダプターは、WebSphere Application Server を統合ブローカーとして使用できるようになりました。詳細については、7 ページの『互換性』を参照してください。 アダプターは、以下のプラットフォーム上で実行されます。

- Solaris 7、8
- AIX 5.x

バージョン 3.2.x

2003 年 3 月更新。「CrossWorlds」という名前は、現在ではシステム全体を表したり、コンポーネント名やツール名を修飾するためには使用されなくなりました。コンポーネント名およびツール名自体は、以前とほとんど変わりません。例えば、「CrossWorlds System Manager」は現在では「System Manager」となり、「CrossWorlds InterChange Server」は「WebSphere InterChange Server」となっています。

このリリースでは、コネクターに対して以下の変更が行われています。

- アダプターは汎用保守フィックスで更新されています。

- 新規データ・ハンドラー (tpi_rnif_mcd2) が追加され、TPI MCD 2.0 フォーマットをサポートするようになりました。このデータ・ハンドラーは CwDataHandler.jar ファイル (ファイル・バージョン 2.3.0) に含まれています。

バージョン 3.1.x

このリリースでは、コネクターに対して以下の変更が行われています。

- コネクターが WebSphere MQ Integrator Broker とともに機能する仕組みを説明するための新しいサンプルが追加されています。
- コネクターが取引先の構成ファイルに対する動的更新をサポートするようになりました。
- コネクターが複数のイベントの並列処理をサポートするようになりました。この機能を実装するために、以下の 4 つのコネクター固有プロパティーが新しく追加されています。
  - DeliverOnArrival: 着信イベントを (PollForEvents モードを使用して) 一度に 1 つずつ処理するか同時に処理するかを指定します。
  - DeliverOnArrivalThreads: 並列イベント処理に使用できるスレッドの数を指定します。
  - WaitForController: コントローラーがアクティブになるまでコネクターが待機する時間を指定します。
  - ControllerWaitCount: コネクターがアクティブでないコントローラーへの接続を試行する回数を指定します。

バージョン 3.0.x

コネクターは国際化されています。詳細については、6 ページの『ロケール依存データの処理』および 31 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。

新規のコネクター固有の構成プロパティー、DataEncoding が追加されて、TPI 文書で使用される文字エンコード方式を指定します。詳細については、13 ページの『コネクター固有のプロパティー』を参照してください。

バージョン 2.1.x

バージョン 2.1.x における新規の変更内容は以下のとおりです。IBM WebSphere Business Integration Adapter for Trading Partner Interchange (TPI) には、TPI コネクターが組み込まれています。このアダプターは、InterChange Server (ICS) および WebSphere MQ Integrator という 2 つの統合ブローカーをサポートしています。統合ブローカーとは、異種のアプリケーション群を統合するアプリケーションのことで、データ・ルーティングなどのサービスを提供します。IBM WebSphere Business Integration Adapter for TPI には、以下が組み込まれています。

- TPI に固有のアプリケーション・コンポーネント
- ビジネス・オブジェクトのサンプル (格納場所は、ディレクトリー ¥connectors¥Tpi¥Samples)
- IBM WebSphere Adapter フレームワーク。コンポーネントは以下のとおりです。
  - コネクター・フレームワーク
  - 開発ツール (Business Object Designer と Connector Configurator を含む)

– API (ODK、JCDK、および CDK を含む)

この資料では、ICS 統合ブローカーと WebSphere MQIntegrator 統合ブローカーの両方を備えたアダプターについて説明します。TPI コネクターのエラー報告機能を改善するために機能が強化されています。

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

バージョン 2.0.x

TPI コネクターのバージョン 2.0.x では、以下の変更が実施されました。

バージョン 1.2.x

TPI コネクターのバージョン 1.2.x では、問題を修正したり、IBM CrossWorlds インフラストラクチャー・バージョン 4.0.0 との互換性を維持するために、小規模な変更が実施されました。

バージョン 1.1.x

IBM CrossWorlds TPI コネクターのバージョン 1.1.x では、TPI バージョン 4.0 をサポートするようになりました。

第 1 章 TPI コネクターの概要

この章は、以下のセクションから構成されています。

コネクターは、*コネクター・フレームワーク* とアプリケーション固有の*コンポーネント* という 2 つの部分から構成されています。コネクター・フレームワークのコードはすべてのコネクターに共通なので、コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントとの仲介役の機能を果たします。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またはテクノロジー (この場合は Trading Partner Interchange) に合わせたコードが格納されています。コネクター・フレームワークは、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・オブジェクトの送信および受信
- 始動メッセージや管理メッセージの交換の管理

この章では、IBM WebSphere Business Integration Trading Partner Interchange (TPI) のコネクター・コンポーネントについて説明します。この資料には、コネクター・フレームワークとアプリケーション固有のコンポーネントの 2 つが説明されていますので注意してください。この資料では、この 2 つのことを、どちらもコネクターと呼んでいます。統合ブローカーとコネクターの関係についての詳細は、「*IBM WebSphere InterChange Server システム管理ガイド*」または「*IBM WebSphere Business Integration WebSphere MQ Integrator Broker 用インプリメンテーション・ガイド*」を参照してください。

TPI Server の概要

このセクションでは、TPI Server の機能について簡単に説明します。コネクターの機能については、その後のセクションで説明します。

TPI Server を使用すると、取引先間でのインターネットを介した文書の交換が安全に実行できます。このアプリケーションでは、取引先間で送信されるセキュア・エンベロープ内の文書が、ユーザーが設定したスケジュールに従ってパッケージ化されます。TPI Server は、XML、EDI、バイナリー・データの各フォーマットをサポートしています。

TPI は、取引先間での文書の転送のために、FTP、SMTP、および WebSphere MQ をサポートしています。TPI Server では、文書の配送や検索のために、ディレクトリーのシステムを採用しています。TPI Server は、取引先ごとに 6 つの文書ディレクトリーを保持します。このうち 3 つは着信用で、残りの 3 つは発信用です。各ディレクトリーには、特定のフォーマット (XML、EDI、バイナリーのいずれか) で記述された着信文書または発信文書が保持されています。

文書の送信

文書がいずれかの発信ディレクトリーに置かれると、TPI Server は、最初にダイジェストを作成して文書をパッケージ化します。このダイジェストには、送信側と受信側の識別番号が記載されています。次に、TPI Server は文書を暗号化して 1 つのメッセージにまとめます。文書のパッケージ化が完了すると、TPI は指定の取引先にこのメッセージを送信します。TPI リポジトリーに格納されている取引先プロファイルは、メッセージの送信に使用する転送方法を判別する役目を果たします。

文書の受信

TPI Server は、取引先からメッセージを受信すると、メッセージを復号して、ディジタル署名を確認し、文書を適切な着信ディレクトリーに書き込みます。メッセージがアンパックされると、TPI Server は、Message Disposition Notification (MDN) をメッセージの送信側に返します。MDN によって確認できることは、メッセージの受信と正常なアンパック化のみです。MDN では、文書が取引先によって十分に処理されたかどうかはわかりません。

TPI コネクター

TPI コネクターを使用すると、統合ブローカーは、TPI Server バージョン 3.0.3 以上との間でビジネス・オブジェクトを交換できるようになります。

コネクターには、ビジネス・オブジェクト処理、イベント・ポーリング、イベント通知機能が実装されています。コネクターのアプリケーション固有のコンポーネントは、ビジネス・オブジェクトを生成して、これを統合ブローカーに送信します。このコンポーネントは、統合ブローカーからのビジネス・オブジェクト要求にも応答します。このコンポーネントは、ファイルやコネクター・コンソールへ書き出すか、または統合ブローカーに送信するロギング・メッセージやトレース・メッセージを生成します。

TPI コネクターは、以下のコンポーネントを使用します。

- WebSphere Business Integration Adapter データ・ハンドラー — コネクターによって呼び出され、TPI Server に対応している文書フォーマットと WebSphere Business Integration Adapter ビジネス・オブジェクトのフォーマット間でフォーマット変換を実行します。
- 取引先の構成ファイル — 取引先ごとのデータ・ハンドラー情報が格納されています。

TPI コネクターは、TPI Server と同じプロセス・スペースで、同じ Java 仮想マシンを使用して動作します。

注: このため、並列処理の度合いに 1 を超える値を設定して TPI コネクターを動作させることはお勧めできません。並列処理の度合いの詳細については、「*システム管理ガイド*」を参照してください。

コネクターは、DocumentListener インターフェースと InterchangeEventListener インターフェースを介して TPI Server と通信します。コネクターは、受け取った TPI 文書を処理してビジネス・オブジェクトを作成し、これを統合ブローカーに渡しま

す。また、要求ビジネス・オブジェクトを処理して、TPI Server へ渡す文書ストリームを作成します。図 1 には、TPI コネクターのアーキテクチャーを示します。

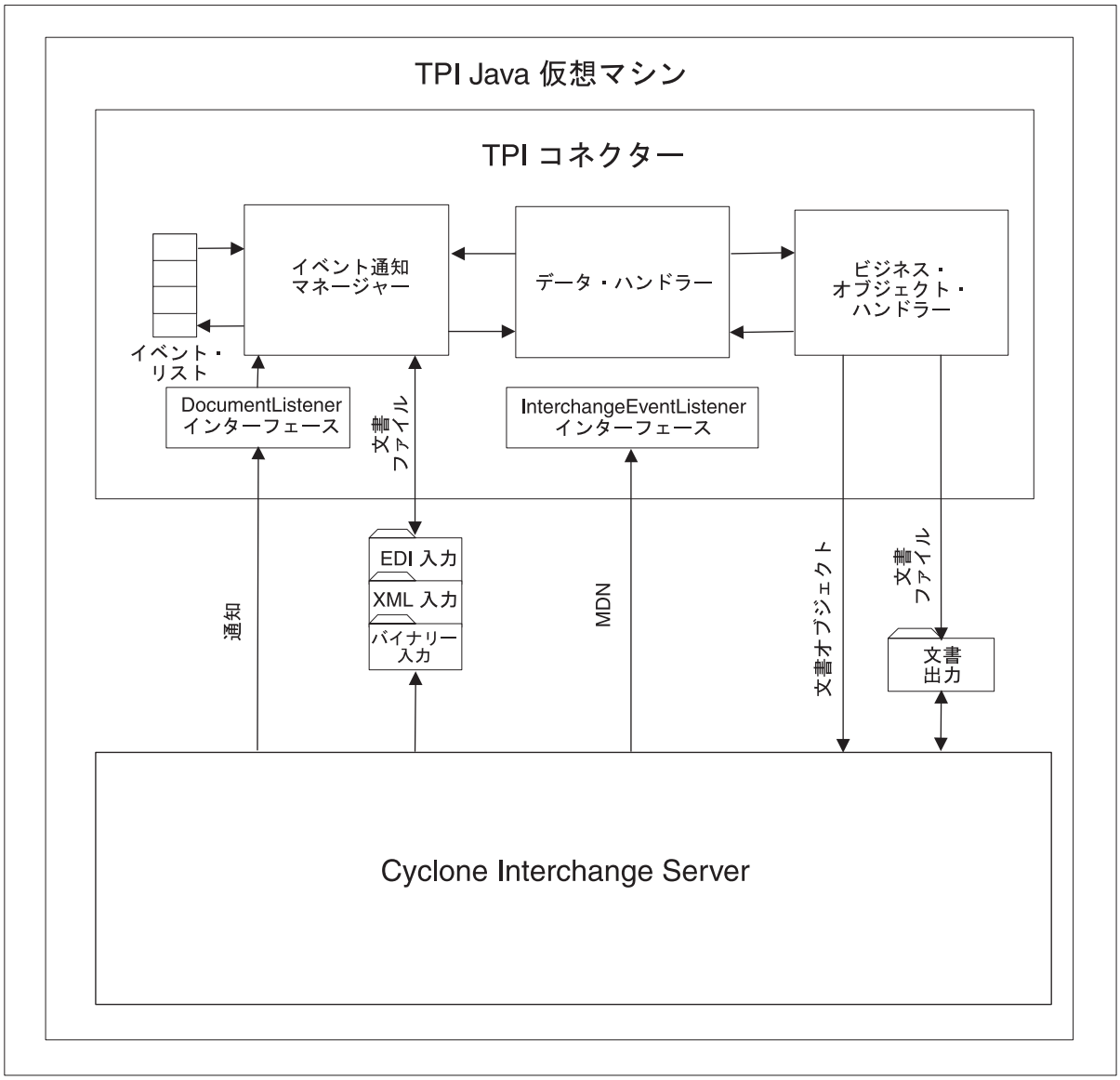

図 1. TPI コネクターのアーキテクチャー

初期化と終了

init() メソッドは、始動時に、コネクター構成プロパティーのロード以外に、以下のタスクを実行します。

- 取引先の構成ファイルを読み込んで、取引先とフォーマットの組み合わせごとにデータ・ハンドラー情報をメモリーにロードする。
- コネクターは、DocmentListener と InterchangeEventListener の 2 つのインターフェースを介して TPI Server およびレジスターを始動し、イベント処理と要求処理を開始する。

- 障害の発生後にコネクターを始動している場合、init() メソッドは、EventRecovery プロパティーで指定したイベント・リカバリー・タスクを実行する。

terminate() メソッドは、コネクターおよび TPI Server をシャットダウンします。

イベント通知

TPI コネクターは、イベント通知時に、ファイル・システムを使用して未処理のイベントのメタデータを永続化します。さらに、処理済みのイベントにイベント状況のラベルを付けてアーカイブ処理します。

TPI コネクターは、イベント通知に対して documentArriving() というコールバック・メソッドを使用します。このメソッドは、DocumentListener インターフェースを介して提供されます。TPI Server は、取引先から文書を受信すると、documentArriving() メソッドを呼び出して、着信文書のコネクターに通知します。この通知には、コネクターが文書の検索や処理を行うために必要なメタデータが格納されています。この内容は次のような情報です。

- 送信側 ID
- 受信側 ID
- 文書タイプ
- 文書ファイル名およびパス
- 固有の文書 ID

これは、イベント・メタデータを構成しています。コネクターは、イベント・リストのイベント・メタデータをメモリーに格納し、.event 拡張子を付けてイベント・ディレクトリーに書き込むことにより、バックアップやリカバリーに備えます。

コネクターは、イベント・リストをポーリングするたびに、コネクターへの到着時刻が最も早いイベントを検索します。コネクターは、In ディレクトリーから文書を検索し、取引先の構成ファイルを使用して文書の MIME タイプを判別します。

コネクターは、MIME タイプを DataHandler クラスに渡して、適切なデータ・ハンドラーのインスタンスを生成します。次に、コネクターはデータ・ハンドラーを呼び出して、文書をビジネス・オブジェクトに変換します。各データ・ハンドラーは、異なる基準を使用してビジネス・オブジェクト名を決定します。ビジネス・オブジェクトが生成されると、コネクターはこのビジネス・オブジェクトへのサブスクリプションの有無を確認します。サブスクリプションが存在する場合、コネクターはサブスクライブ側のビジネス・プロセスにこのビジネス・オブジェクトを渡します。(使用している統合ブローカー専用のサブスクリプション情報については、統合ブローカーのインプリメンテーション・ガイドを参照してください。)

並列処理

デフォルトでは、TPI コネクターは一度に 1 つのイベントしか処理しません。しかし、複数の着信イベントを同時に処理し、パフォーマンスを向上させることができます。この場合、各イベントには固有のスレッドが割り振られ、そのスレッドが同時に実行されます。

並列処理を使用可能にするには、コネクター固有のプロパティーを使用します。詳細については、 13 ページの『コネクター固有のプロパティー』を参照してください。

イベント・リカバリー

コネクターは、イベント・リストからイベントを取り込むと、処理が完了するまで、つまり、イベント・ファイルがアーカイブ・ディレクトリーに移動する時刻まで、このイベントに対応するイベント・ファイルの拡張子を .inprogress に変更します。コネクターの障害発生後にコネクターを再始動すると、コネクターは、すべての `inprogress` イベントを EventRecovery プロパティーの設定値に従って処理します。EventRecovery を `Reprocess` に設定すると、コネクターはイベント・ディレクトリーを読み取って、すべての `inprogress` イベントをイベント・リストに格納します。コネクターの構成によっては、`inprogress` イベントを無視するか、このイベントが失敗するように設定することもできます。

注: コネクターが `documentArriving` イベント通知を受信してから、イベントをファイルに書き込むまでのわずかな時間の間に、コネクターを突然終了すると、コネクターはこのイベントを回復できません。

イベントのアーカイブ処理

コネクターは、コネクターが処理済みイベントを格納する場所である archive ディレクトリーを保守します。イベントが処理されると、イベント・ファイルの名前が変更され、次のいずれかの拡張子が付けられます。

- archive - イベントはコネクターによって正常に処理されました。
- unsubscribed - イベントに対するサブスクリプションは存在しません。使用している統合ブローカー専用のサブスクリプション情報については、統合ブローカーのインプリメンテーション・ガイドを参照してください。
- fail - コネクターは、イベントを処理して統合ブローカーに通知することができません。

ビジネス・オブジェクト要求の処理

TPI コネクターによって処理されたすべての要求ビジネス・オブジェクトには、必ず構成の子オブジェクトが含まれています。このオブジェクトには、コネクターや TPI Server によってオブジェクトを処理するためにコネクターによって要求された情報が格納されています。これには、送信側 ID、受信側 ID、および文書タイプが含まれます。

コネクターがビジネス・オブジェクトを受信すると、doVerbFor() メソッドはデータ・ハンドラーを呼び出して、ビジネス・オブジェクトを、適切なフォーマット済みストリームに変換します。データ・ハンドラーの呼び出しは、ReceiverID の値や DocumentType の値については、取引先の構成ファイルに登録されている MIME タイプが基準になります。TPI は、SenderID、ReceiverID および CycloneID について、EDI 文書および XML 文書を解析します。データ・ハンドラーは、文書出力ディレクトリーのファイルに書き込まれている文書ストリームを出力します。コネクターは、TPI Server API の sendDocument() メソッドを呼び出すことによって、このファイルを参照する文書オブジェクトを TPI Server に渡します。sendDocument() メソッドは、TPI Server が生成した固有の文書 ID を返します。

WaitForMDN コネクターのプロパティーが true に設定されている場合、コネクターは、TPI Server が取引先から MDN を受信したことを表示するまで待機します。子メタオブジェクトの WaitForMDN 属性を取り込むと、WaitForMDN プロパティーの指定をビジネス・オブジェクトごとに変更することもできます。MDN が受信されると、doVerbFor() メソッドは、統合ブローカーに状況コード (成功またはエラー) を返します。WaitForMDN プロパティーが false に設定されている場合、コネクターは、TPI Server が MDN を返すのを待機しません。

ロケール依存データの処理

コネクターは国際化されているので、2 バイト文字セットに対応でき、指定された言語でメッセージ・テキストを配送できます。コネクターがある文字コード・セットを使用する場所から別の文字コード・セットを使用する場所にデータを転送する場合、文字変換を行ってデータの意味を保持します。

Java 仮想マシン (JVM) 内の Java ランタイム環境では、データを Unicode 文字コード・セットで表現します。Unicode は周知の文字コード・セット (単一バイトとマルチバイトの両方) で文字をエンコードします。 WebSphere Business Integration システムのほとんどのコンポーネントは、Java で記述されています。したがって、ほとんどの WebSphere Business Integration システムのコンポーネントの間でデータが転送されても、文字変換の必要はありません。

該当する国または地域の適切な言語でエラーおよび情報メッセージをログに記録するには、使用する環境の Locale 標準構成プロパティーを設定します。これらのプロパティーの詳細については、 31 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

第 2 章 コネクターのインストールと構成

この章では、IBM WebSphere Business Integration Adapter for Trading Partner Interchange (TPI) のコネクター・コンポーネントをインストールして構成する方法について説明します。この章は、以下のセクションから構成されています。

互換性

アダプターが使用するアダプター・フレームワークは、アダプターと通信する統合ブローカーのバージョンとの互換性を備えている必要があります。Adapter for Trading Partner Interchange のバージョン 3.4 は、以下の統合ブローカーを使用する以下のバージョンのアダプター・フレームワークでサポートされます。

アダプター・フレームワーク: WebSphere Business Integration Adapter Framework バージョン 2.1、2.2、2.3.x、および 2.4。

統合ブローカー:

- WebSphere InterChange Server バージョン 4.1.1、4.2.0、4.2.1、4.2.2
- WebSphere MQ Integrator バージョン 2.1.0
- WebSphere MQ Integrator Broker バージョン 2.1.0
- WebSphere Business Integration Message Broker バージョン 5.0

例外については、「リリース情報」を参照してください。

注: 統合ブローカーのインストール手順およびその前提条件については、次の資料を参照してください。WebSphere InterChange Server (ICS) については、「システム・インストール・ガイド (UNIX 版)」または「システム・インストール・ガイド (Windows 版)」を参照してください。

Message Brokers (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) の場合は、「*WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド*」およびそれぞれの Message Brokers のインストールに関する資料を参照してください。一部の資料は次の Web サイトにあります。
http://www.ibm.com/software/integration/mqfamily/library/manualsa/

WebSphere Application Server については「*アダプター実装ガイド (WebSphere Application Server)*」および次の場所にある資料を参照してください。
http://www.ibm.com/software/webservers/appserv/library.html

前提条件

このセクションでは、必須のソフトウェア・コンポーネントと、コネクターをインストールする前に実行するタスクについて説明します。

TPI Server のインストール

Trading Partner Interchange (TPI) Adapter バージョン 3.4.x は、Trading Partner Interchange Server バージョン 4.1 および 4.2 とともに使用することがサポートされています。

TPI Server は、必ず TPI コネクターをインストールする前にインストールしてください。TPI コネクターと TPI Server は、同じマシンにインストールする必要があります。TPI Server のインストールおよび構成の手順については、TPI Server に付属の「*Administrator's Guide*」を参照してください。

Solaris でのアカウントとアクセス権のセットアップ

Solaris システムに TPI をインストールする場合は、あらかじめ WebSphere Business Integration Adapter と TPI Server の管理者をメンバーにした上で、ユーザー・グループを作成します。この作業は、TPI コネクターが TPI Server にアクセスできるようにするためには必須です。

どちらのグループのメンバーにも、TPI Server のインストール・ディレクトリーに対する読み取り/書き込み許可が与えられていることを確認してください。

データベース・サポート

TPI Server には、Sybase SQL Anywhere のランタイム・バージョンが標準のリポジトリーとして組み込まれています。これはランタイム・バージョンなので、データベースの管理や表示に関するアクセス権は限定されています。したがって、Oracle か Microsoft Server SQL Server のいずれかを TPI Repository として構成することをお勧めします。外部データベースを TPI Repository として構成する方法については、TPI Server に付属の「*Administrator's Guide*」を参照してください。

TPI アダプターおよび関連ファイルのインストール

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

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

TPI Server イベント・ロギング機能の構成

TPI コネクターは、TPI Server によって記録されたイベント・メッセージを使用して、イベントの処理をモニターします。TPI Server は、必ず「Debug」モードで稼働するように構成する必要があります。こうすると、TPI Server は、すべてのサーバー・イベントをコネクターに通知するようになります。

注: TPI Server が「Debug」モードで稼働していないと、コネクターは、TPI Server `init()` メソッドがイベント完了メッセージを待機している間に停止します。

TPI Server が「Debug」モードで稼働するよう設定するには、次の手順に従います。

1. TPI Server Administrator を始動する。
2. 「Administrator」メニューの「ツール」>「設定」を選択する。
3. 「General」タブで、「Event Logging Level」を「Alert」、「Notify」、「Transaction」、「Debug」に設定する。

コネクターの構成方法

注: TPI コネクターと TPI Server は同じアドレス・スペースで動作するため、並列処理の度合いに 1 を超える値を設定して TPI コネクターを動作させることはお勧めできません。並列処理の度合いの詳細については、「*システム管理ガイド*」を参照してください。

コネクターを始動する前に、以下の構成作業を実行する必要があります。

- 取引先の構成ファイルを作成する。
- コネクターを構成して、TPI Server を始動する。
- コネクターの構成プロパティーを設定する。

取引先の構成ファイルの作成

取引先の構成ファイルには、XML、EDI、バイナリーの各フォーマットごとに、各取引先が使用している MIME タイプが登録されています。コネクターは、文書またはビジネス・オブジェクトを処理するたびに、取引先の構成ファイルから MIME タイプを読み込みます。この MIME タイプは、対象の文書またはビジネス・オブジェクトに対して正しいデータ・ハンドラーを呼び出すために、DataHandler クラスに渡されます。

取引先の構成ファイルは、タブ区切りのテキスト・ファイルです。このファイルの各行には、TPI 取引先 ID と、その後に、XML、EDI、バイナリーの各フォーマットごとに、取引先が使用している MIME タイプが記述されています。取引先の構成ファイルのフォーマットは、次のとおりです。

取引先 ID <タブ> XML MIME タイプ <タブ> EDI MIME タイプ <タブ> バイナリー MIME タイプ

フォーマット設定が適切であることを確認するため、Microsoft の Excel シートとしてファイルを作成します。次に、テキスト (タブ区切り) としてファイルを保存します。図 2 は、Excel でファイルを作成する方法を示しています。

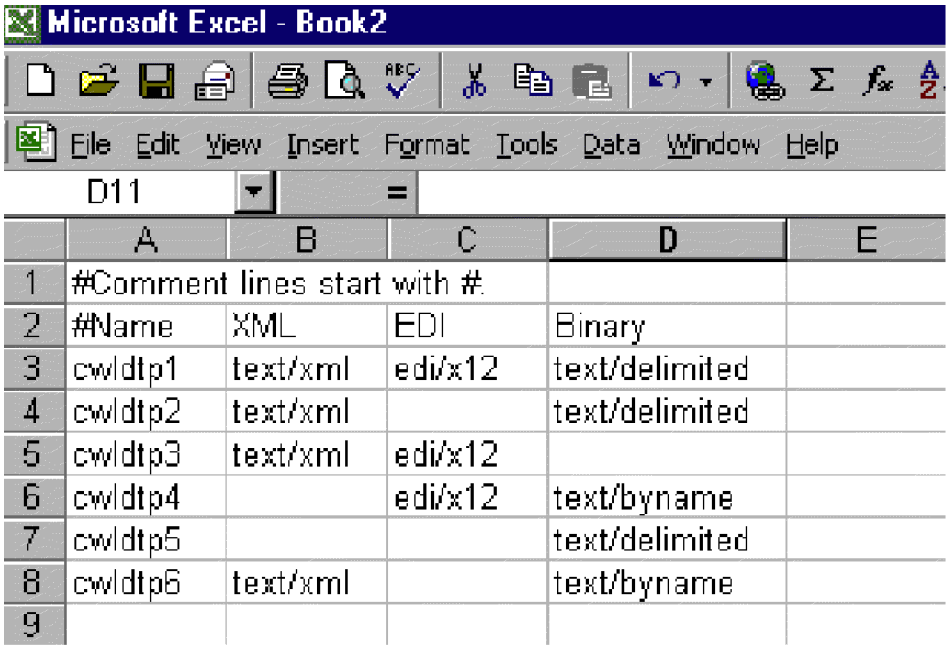

図 2. Excel による取引先の構成ファイルの作成

このファイルは、TPI をインストールしたマシンの任意のディレクトリーに保存できます。TradingPartnerConfigurationFile というコネクター・プロパティーには、このファイルの位置情報が保持されています。詳細については、13 ページの『コネクター固有のプロパティー』を参照してください。

コネクターの実行中に、取引先の構成ファイルを更新して新規取引先を追加したり文書情報を変更したりすることができます。更新内容をロードするためにコネクターを再始動する必要はありません。コネクターは、処理中に構成ファイルから更新内容を取得します。

TPI Server を始動するためのコネクターの構成

TPI コネクターは、TPI Server を実行するために TPI Server とともに提供される JVM を使用します。この JVM を使用することにより、TPI コネクターが Java プロセスとして開始し、このプロセスの中から TPI Server を始動します。この動作は、コネクターのインストール後に必ず構成してください。構成するには、コネクターの始動ファイル (Windows NT では `start_TPI.bat`、Solaris では `start_TPI.sh`) を編集します。TPI Server の始動およびシャットダウンを行うためにコネクターを構成するには、次の手順を実行します。

1. `¥connectors¥TPI` ディレクトリーに置かれている `start_TPI` ファイルを開きます。
2. `CYCLONEHOMEDIR` 属性の値を TPI のホーム・ディレクトリー・パスに変更します。
3. ファイルを保存して閉じます。

TPI Adapter が正常に TPI Server のインスタンスを生成できるようにするためには、TPI Server と同時に提供された JVM も構成する必要があります。TPI が提供した JVM を構成するには、次の手順に従います。

Windows の場合:

1. TPI Adapter のディレクトリー: %CROSSWORLDS%¥connectors¥TPI 内で ¥dependencies¥win フォルダーの場所を探します。
2. TPI ホーム・ディレクトリーの中の ¥cijre フォルダーの場所も探します。
3. %CROSSWORLDS%¥connectors¥TPI¥dependencies¥win フォルダーの内容を %CYCLONEHOMEDIR%¥cijre フォルダーにコピーします。
4. 以下のファイルが %CYCLONEHOMEDIR%¥cijre に置かれました。
   - ¥bin¥orb.dll
   - ¥bin¥orb.dll
   - ¥lib¥orb.properties
   - ¥lib¥ext¥ibmorb.jar
   - ¥lib¥ext¥ibmext.jar

Solaris の場合:

1. TPI Adapter のディレクトリー: ${CROSSWORLDS}/connectors/TPI 内で /dependencies/sol フォルダーの場所を探します。
2. TPI Server のホーム・ディレクトリーの中の /cijre フォルダーの場所も探します。これは ${CYCLONEHOMEDIR} として識別されます。
3. ${CROSSWORLDS}/connectors/TPI/dependencies/sol フォルダーの内容を ${CYCLONEHOMEDIR}/cijre フォルダーにコピーします。
4. 以下のファイルが ${CYCLONEHOMEDIR}/cijre に置かれました。
   - ¥lib¥orb.properties
   - ¥lib¥ext¥ibmorb.jar
   - ¥lib¥ext¥ibmext.jar
   - ¥lib¥sparc¥liborb.so
   - ¥lib¥sparc¥liborb_g.so

始動ファイルの編集に加えて、コネクターが正常に TPI Server を始動できるように、以下のコネクター・プロパティーを設定する必要があります。

- MetaEventDir
- DocumentOutDir
- TradingPartnerConfigurationFile
- ArchiveProcessedDocDirSee

詳細については、13 ページの『コネクター固有のプロパティー』を参照してください。

Windows Service としてのコネクターと TPI Server の実行

TPI コネクターと TPI Server は、Windows Service として実行できます。コネクターをサービスとしてインストールしたら、start_TPI.bat ファイルも次のように変更する必要があります。

- コネクターを Cyclone JVM から実行するために、*%ProductDir%*¥bin¥java の値を %CYCLONEHOMEDIR%¥cijre¥bin¥java で置き換える。

- 次の記述を %CYCLONEHOMEDIR%¥cijre¥bin¥java コマンド行の末尾に追加して、JRE の出力の宛先をファイルに変更する。

```
>"%CONNDIR%"¥connectors¥TPI¥TPITrace.txt
```

コネクター構成プロパティーの設定

コネクターを実行するには、その前に、コネクターの標準構成プロパティーとコネクター固有の構成プロパティーを設定する必要があります。コネクターの構成プロパティーを設定するには、次のいずれかのツールを使用します。

- Connector Configurator (WebSphere ICS が統合ブローカーの場合): このツールには、System Manager からアクセスします。
- Connector Configurator (WebSphere MQ Integrator Broker が統合ブローカーの場合): このツールには、IBM WebSphere Business Integration Adapter のプログラム・フォルダーからアクセスします。Connector Configurator の詳細については、51 ページの『付録 B. Connector Configurator』を参照してください。

標準コネクター・プロパティー

標準の構成プロパティーは、すべてのコネクターが使用する情報を提供します。これらのプロパティーの詳細については、31 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。

重要

コネクターは ICS と WebSphere MQ Integrator Broker の 2 つの統合ブローカーをサポートしているので、この 2 つの統合ブローカーの構成プロパティーは、コネクターに関係があります。

また、IBM WebSphere Business Integration Adapter for TPI に固有の構成情報については、表 1を参照してください。この表に示す情報は、付録の補足です。

表 1. このコネクターに固有のプロパティー情報

プロパティー	注
AgentConnections	このコネクターは、単一スレッド化されています。したがって、AgentConnections プロパティーを使用することはできません。
CharacterEncoding	このコネクターは Java ベースなので、CharacterEncoding プロパティーを使用しません。

表 1. このコネクターに固有のプロパティー情報 (続き)

プロパティー	注
Locale	このコネクターは国際化されているので、Locale プロパティーを変更できます。現在サポートされているロケールを判別するには、コネクターのリリース情報を参照してください。 **注:** 統合ブローカーとして WebSphere MQ Integrator Broker を使用している場合、アダプター、ブローカー、およびすべてのアプリケーションに同じロケールを使用する必要があります。

コネクター固有のプロパティー

コネクター固有の構成プロパティーには、コネクターが実行時に必要な情報が記述されています。コネクター固有のプロパティーには、コネクターのコードを再記述したり再ビルドしたりすることなく、コネクター内部の構成情報や論理を変更する方法も記載されています。

表 2 には、コネクターに関するコネクター固有のプロパティーを示します。プロパティーの説明については、以下の各セクションを参照してください。

表 2. 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
ArchiveProcessedDocDir	*<有効なディレクトリー・パス>*	
ArchiveProcessedDocInfo	`true` または `false`	true
BackupRequired	`true` または `false`	true
CycloneServerArgs	コネクターが使用している引き数と競合しない、任意の有効な TPI Server 引き数。	-appagent; -nogui; -console;
DataEncoding	JVM でサポートされる有効なエンコード名。	このプロパティーに値が指定されていない場合は、デフォルトでオペレーティング・システムによって設定されたエンコード・タイプになります。
DataHandlerDefaultMO		MO_DataHandler_ Default
DefaultBinaryMimeType	*<有効な MIME タイプ>*	
DefaultEDIMimeType	*<有効な MIME タイプ>*	
DefaultVerb	コネクターがサポートしているすべての動詞。	
DefaultXMLMimeType	*<有効な DataHandler クラス名>*	
DeliverOnArrival	`true` または `false`	false
DeliverOnArrivalThreads	1 から 10 の任意の数値。	1

表 2. 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
DocumentOutDir	*<有効なディレクトリー・パス>*	
EventRecovery	`FailOnStartup`、`Reprocess`、`LogError`、および `Ignore`	FailOnStartup
MetaEventDir	*<有効なディレクトリー・パス>*	
PollQuantity		25
TradingPartnerConfigurationFile	*<完全修飾ファイル名>*	
WaitForController		500 ミリ秒
WaitForMDN	`true` または `false`	true

ArchiveProcessedDocDir: 処理済み文書のメタイベントが保存されるディレクトリー。ArchiveProcessedDocInfo プロパティーを true に設定した場合は、このプロパティーは必須です。

ArchiveProcessedDocInfo: 文書が正常に処理され、統合ブローカーへ送信された後、コネクターがメタイベントを保存するかどうかを指定します。このプロパティーを false に設定すると、メタイベントは処理後に削除されます。デフォルト値は true です。

BackupRequired: TPI Server が各文書の送信後、文書をバックアップするかどうかを指定します。true に設定すると、TPI は、各文書を送信後、文書をバックアップします。デフォルト値は true です。

ControllerWaitCount: コントローラーがアクティブかどうかをコネクターが確認する最大回数を指定します。このプロパティーは WaitForController プロパティーとともに使用します。コントローラーがアクティブかどうかをコネクターが確認するたびに、WaitForController で指定された時間だけ待機します。試行回数が最大回数に達してもコントローラーが応答しない場合は、イベント処理は失敗します。デフォルト値は 1 です。

CycloneServerArgs: 始動時に TPI Server に渡される対象の引き数のセミコロン区切りのリスト。このプロパティーは必須です。デフォルト値は、-appagent; -nogui; -console; です。

DataEncoding: コネクターが使用するデータ・エンコードのタイプを指定します。有効な値には、JVM でサポートされるエンコード・タイプなどがあります。非 ASCII 文字エンコード方式が使用される場合には、必ずこのプロパティーを設定する必要があります。DataEncoding プロパティーを設定しない場合は、コネクターは、オペレーティング・システム・レベルで指定されたエンコード・タイプを使用します。

DataHandlerDefaultMO: コネクターのメタオブジェクトのデフォルトの名前。これは、サポートされているフォーマットごとにデータ・ハンドラーの構成を決定するために使用する名前です。このプロパティーは必須です。デフォルト値は、MO_DataHandler_Default です。

DefaultBinaryMimeType: 取引先の構成ファイルに取引先が設定されていない場合、データ・ハンドラーがバイナリー文書を使用するための MIME タイプ。TPI にバイナリー MIME タイプが使用されている場合、このプロパティーは必須です。このプロパティーには、デフォルト値は設定されていません。

DefaultEDIMimeType: 取引先の構成ファイルに取引先が設定されていない場合、データ・ハンドラーが EDI 文書を使用するための MIME タイプ。TPI に EDI MIME タイプが使用されている場合、このプロパティーは必須です。このプロパティーには、デフォルト値は設定されていません。

DefaultVerb: ポーリング時にデータ・ハンドラーによって動詞が設定されなかった場合、着信ビジネス・オブジェクトの内部に設定する動詞を指定します。

DefaultXMLMimeType: 取引先の構成ファイルに取引先が設定されていない場合、データ・ハンドラーが XML 文書を使用するための MIME タイプ。TPI に XML MIME タイプが使用されている場合、このプロパティーは必須です。このプロパティーには、デフォルト値は設定されていません。

DeliverOnArrival: 使用するイベント処理の方式を指定します。DeliverOnArrival を false に設定した場合は、イベントは一度に 1 つずつ処理されます。すなわち PollForEvent スレッドがメモリーからイベントを取得し、そのイベントを処理してから次のイベントを取得します。DeliverOnArrival を true に設定すると、複数のイベントを同時に処理できます。並列に処理できるイベントの数は DeliverOnArrivalThreads プロパティーで決定されます。DeliverOnArrival のデフォルト値は false です。

DeliverOnArrivalThreads: 同時イベント処理に割り振るスレッドの数を指定します。DeliverOnArrival プロパティーを true に設定する場合は、DeliverOnArrivalThreads プロパティーを着信イベントに割り振るスレッドの数に設定してください。並列処理の最小値は 2 であり、最大値は 10 です。デフォルトでは、DeliverOnArrivalThreads は 1 に設定されており、並列処理は使用不能になっています。

DocumentOutDir: 発信文書を TPI が処理するまで、この発信文書の一時的な書き込み先となるディレクトリーの位置。システム障害が発生した場合、文書はこのディレクトリーから回復できます。このプロパティーは必須です。

EventRecovery: イベント・リカバリーの動作を指定します。コネクターは、再始動時に出力ディレクトリーを調べて、.inprogress という拡張子の付いたファイルの有無を確認します。このプロパティーを FailOnStartup に設定すると、コネクターは始動できなくなります。Reprocess に設定すると、コネクターはこれらのイベントをサーバーに再発信します。LogError に設定すると、コネクターはエラーを記録しますが、シャットダウンはしません。Ignore に設定すると、コネクターはこうしたイベントを無視します。デフォルト値は FailOnStartup です。

MetaEventDir: リカバリーを目的として TPI イベント情報を永続化するときに使用するディレクトリー。このプロパティーは必須です。

PollQuantity: コネクターがイベント・リストをポーリングするたびに検索するイベントの数を指定します。デフォルト値は、25 です。

TradingPartnerConfigurationFile: 取引先の構成ファイルの完全修飾名。このファイルには、バイナリー、XML、EDI の各メッセージに関して各取引先から受け取った文書に使用されている MIME タイプが記述されています。MIME タイプは、文書ごとに正しいデータ・ハンドラーを呼び出すために使用します。このプロパティーは必須です。このプロパティーが指定されていないと、コネクターは始動できません。

WaitForController: コントローラーがアクティブになるまでコネクターが待機する時間をミリ秒単位で指定します。このプロパティーは ControllerWaitCount プロパティーとともに使用します。デフォルト値は 500 ミリ秒です。

WaitForMDN: 文書を TPI Server に渡した後、コネクターが取引先からの MDN を待機するか、または要求スレッドから戻すかを指定します。MDN には、プロトコル・レベルでは送信が正常に開始された、ということが記述されています。子メタオブジェクトの WaitForMDN 属性を設定すると、このプロパティーの設定をビジネス・オブジェクト単位で変更できます。デフォルト値は true です。

複数のコネクター・インスタンスの作成

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

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

新規ディレクトリーの作成

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

```
ProductDir¥connectors¥connectorInstance
```

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

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

```
ProductDir¥repository¥connectorInstance
```

ビジネス・オブジェクト定義の作成

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

1. 初期コネクターに関連付けられているビジネス・オブジェクト定義を変更する必要がある場合は、適切なファイルをコピーし、Business Object Designer を使用

してそれらのファイルをインポートします。初期コネクターの任意のファイルをコピーできます。変更を加えた場合は、名前を変更してください。

2. 初期コネクターのファイルは、次のディレクトリーに入っていなければなりません。

   ```
   ProductDir¥repository¥initialConnectorInstance
   ```

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

コネクター定義の作成

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

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

始動スクリプトの作成

始動スクリプトは以下のように作成します。

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

   dirname

2. この始動スクリプトを、16 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. 始動スクリプトのショートカットを作成します (Windows のみ)。
4. 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。

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

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

コネクターの始動

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

ProductDir¥connectors¥*connName*

ここで、*connName* はコネクターを示します。始動スクリプトの名前は、表 3 に示すように、オペレーティング・システム・プラットフォームによって異なります。

表 3. コネクターの始動スクリプト

オペレーティング・システム	始動スクリプト
UNIX ベースのシステム	`connector_manager_connName`
Windows	`start_connName.bat`

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

- Windows システムで**「スタート」**メニューから。

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

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

    `connector_manager_connName -start`

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

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

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

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

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

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

コネクター始動のための特殊な考慮事項

Cyclone 4.1 以前を使用している場合は、WebSphere MQ Integrator Broker をブローカーとして使用する場合に (WebSphere MQ Integrator Broker 統合ブローカーの場合のみ)、TPI 用コネクターの始動に関する問題が発生する場合があります。ローカルのコネクター構成ファイルに暗号化されたプロパティー値 (例えばパスワード) が含まれていると、始動時に java.lang.NullPointerException という例外が発生して、コネクターは終了してしまいます。

この問題を防止するには、java.security ファイルのプロバイダー・リストを次の値で置き換えます。

```
security.provider.1=com.cyclonecommerce.crossworks.provider.Cyclone
security.provider.2=iaik.security.provider.IAIK
security.provider.3=sun.security.provider.Sun
```

Windows システムと Solaris システムの場合、ファイルは %CYCLONEHOMEDIR%/cijre/lib/security ディレクトリーに置かれています。

コネクターの停止

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

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

    ```
    connector_manager_connName -stop
    ```

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

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

このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。

- Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムのシャットダウン時に、コネクターは停止します。

第 3 章 コネクターのビジネス・オブジェクトの開発

この章では、Trading Partner Interchange (TPI) ビジネス・オブジェクトに必要な要素と構造について説明します。特に、TPI コネクターのビジネス・オブジェクト要件について具体的に説明します。この章は、以下のセクションから構成されています。

データ・ハンドラーおよび文書のフォーマット

注: Adapter for TPI は、Datahandler for RNIF over TPI にも付属しています。 79 ページの『付録 D. Adapter for Trading Partner Interchange のサンプル・シナリオ』を参照してください。

コネクターは、XML、EDI、およびバイナリーの各フォーマットの文書交換をサポートしています。TPI コネクターは、データ・ハンドラーを使用して TPI 文書を WebSphere Business Integration Adapter のビジネス・オブジェクトに変換し、WebSphere Business Integration Adapter のビジネス・オブジェクトを TPI 対応の文書フォーマットに変換します。

表 4. WebSphere Business Integration Adapter に付属のデータ・ハンドラー

文書フォーマット	WebSphere Business Integration Adapter に付属のデータ・ハンドラー
XML	XML
EDI	EDI
バイナリー	任意

コネクターは、文書の MIME タイプを DataHandler クラスに渡すことによって、適切なデータ・ハンドラーを呼び出します。さらに、コネクターは、BOPrefix の値が着信側のビジネス・オブジェクトに取り込まれている場合、この値も渡します。受け取った文書を処理する場合、コネクターは、取引先の構成ファイルからこの文書の MIME タイプを読み取ります。発信側の要求ビジネス・オブジェクトを処理する場合は、子メタオブジェクトの DocumentType 属性と ReceiverID 属性の値に基づいて、取引先の構成ファイルから MIME タイプを読み取ります。詳細については、 23 ページの『ビジネス・オブジェクトの構造』を参照してください。

ビジネス・オブジェクトのデータ・ハンドラー要件

TPI コネクターは、ストリームとビジネス・オブジェクトとの双方向の変換を実行するときにデータ・ハンドラーを呼び出すため、各ビジネス・オブジェクトは、変換を実行するために呼び出されたデータ・ハンドラーの指定内容に合致している必要があります。特定のデータ・ハンドラーに関する具体的なビジネス・オブジェクトの詳細については、「データ・ハンドラー・ガイド」を参照してください。

XML 文書および EDI 文書に対する TPI Server の要件

TPI Server には、EDI 文書および XML 文書のヘッダー情報の内容とフォーマット設定に対して、固有の要件があります。すべての XML 文書および EDI 文書には、有効な SenderID、RecieverID および固有の CycloneID が記述されている必要があります。SenderID または ReceiverID の値が無効な場合、または CycloneID の値が TPI システム内で固有ではない場合、TPI Server は対象の文書を処理しません。データ・ハンドラーがこれらの値を文書内に正しく配置できるように、これらの値は要求ビジネス・オブジェクトに対応する必要があります。

EDI の要件

EDI 文書の場合、これらの値の配置は、EDI 仕様によって義務付けられています。次の EDI ヘッダーのサンプルについて考えてみましょう。

```
ISA*00* *01*XXXXXX *L1*2 *L0*0*961106*2106*U*00302*000087875*
```

SenderID: SenderID は 2 つの列 (この例では *L1*2) から読み取ります。最初の列には、2 文字の SenderID 修飾子が必要です。2 番目の列には、SenderID の残りのすべての文字が必要です。

ReceiverID: SenderID と同様に、ReceiverID も EDI 文書ヘッダーの 2 つの列 (この例では *L0*0) から読み取ります。最初の列には、2 文字の ReceiverID 修飾子が必要です。2 番目の列には、ReceiverID の残りのすべての文字が必要です。

CycloneID: EDI 文書では、CycloneID は EDI Control ID に対応します。CycloneID は、1 列 (この例では *000087875*) で表されます。この値は、EDI Control ID 番号である 87875 に対応しています。

XML の要件

XML 文書での SenderID、ReceiverID、CycloneID の配置は、TPI Server Administrator を使用してカスタマイズできます。詳細については、TPI Server に付属の「Administrator's Guide」を参照してください。

ビジネス・オブジェクトおよび属性の命名規則

要求処理に使用するビジネス・オブジェクトには、特定の命名規則はありません。イベント通知で処理されるビジネス・オブジェクトは、受け取った文書のフォーマットと WebSphere Business Integration Adapter のビジネス・オブジェクト間の変換を実行するデータ・ハンドラーの命名規則に従っていることが必要です。詳細については、「データ・ハンドラー・ガイド」を参照してください。

ビジネス・オブジェクトの構造

TPI コネクターには、コネクター自身が処理するビジネス・オブジェクトについて 2 つの要件が存在します。

- 各ビジネス・オブジェクトには、コネクターが必要とする動的なメタデータを保持しているメタオブジェクトが格納されている必要があります。これは、TPI Server の API を使用して、内容のフォーマット設定や文書の送信を行うためです。
- 各ビジネス・オブジェクトは、ビジネス・オブジェクトをデータ・ストリームに変換するときに使用するデータ・ハンドラーが必要とするビジネス・オブジェクトの構造に合致する必要があります。

子メタオブジェクトの属性

ビジネス・プロセスからコネクターに送信される各ビジネス・オブジェクトには、基数が 1 の子として基数が 1 つのメタオブジェクトが存在する必要があります。このメタオブジェクトには、コネクターがビジネス・オブジェクトを変換するために適切なデータ・ハンドラーを呼び出したり、TPI Server の API にある sendDocument() メソッドを呼び出したりするときに必要となる動的なメタデータが格納されています。表 5 には、メタオブジェクトの属性を示します。この表に記載されているすべての属性は、ビジネス・オブジェクト定義で定義しておく必要があります。

表 5. TPI コネクターのメタオブジェクト属性

属性名	説明	デフォルト値
DocumentExt	文書のファイル拡張子を指定します。要求処理時に使用されます。	
DocumentType	文書のフォーマット (XML、EDI、またはバイナリー) を指定します。	binary
BOPrefix	MIME タイプとともに、XML データ・ハンドラーのインスタンス作成に使用されます。	
SenderID	文書を送信している取引先の固有 TPI ID を指定します。	デフォルト値はユーザーが設定する必要があります。
ReceiverID	文書を受信している取引先の固有 TPI ID を指定します。	デフォルト値はユーザーが設定する必要があります。
UniqueID	TPI Server によって各文書に割り当てられた固有 ID を指定します。この属性はオプションですが、ビジネス・オブジェクト要求の処理に使用できます。	
OriginalName	文書出力ディレクトリーに書き込む文書オブジェクト・ファイルに名前を付けるときに使用する接頭部を指定します。	ユーザーが設定する必要があります。
WaitForMDN	文書の送信後、コネクターが TPI Server からの MDN を待機するかどうかを指定します。	true
BackupRequired	TPI Server が文書を取引先に送信後、文書のバックアップ・コピーを作成するかどうかを指定します。	true

DocumentExt

DocumentExt 属性により、要求処理時のファイル拡張子 (.xml や .edi など) を指定できます。これはオプションの属性であり、デフォルト値はありません。

DocumentType

DocumentType 属性は、取引先の構成ファイルから文書の MIME タイプを読み取るときに、ReceiverID 属性と組み合わせて使用します。コネクターは、MIME タイプを使用して適切なデータ・ハンドラーを呼び出します。デフォルト値は binary です。

DocumentType 属性には、CW_RNIF という特殊な値も使用できます。この状況では、ReceiverID 属性に値を指定する必要はありません。コネクターはこの属性を使用しないからです。

BOPrefix

BOPrefix は、適切なデータ・ハンドラーのインスタンスを呼び出すために、コネクターによって MIME タイプと組み合わせて使用されます。データ・ハンドラーのインプリメンテーション環境で処理される MIME タイプが 1 つのみの場合、子メタオブジェクトの BOPrefix 属性はオプションになります。

SenderID

SenderID は、文書送信側の固有な取引先 ID です。この値は DefaultDocument オブジェクトを作成するためにコネクターによって使用されます。この値は、sendDocument() 呼び出しによって渡されます。

ReceiverID

ReceiverID は、文書の送信先となる取引先の固有 ID です。この値は、取引先の構成ファイルから文書の MIME タイプを読み取るときに、DocumentType 属性と組み合わせて使用します。この値は DefaultDocument オブジェクトを作成する目的でも使用され、sendDocument() 呼び出しによって渡されます。この属性はオプションですが、ビジネス・オブジェクトの処理に使用できます。

UniqueID

TPI Server によって各文書に割り当てられた固有 ID。

Original Name

コネクターが TPI Server による検索のために DocumentOutDir ディレクトリーに書き込む出力ファイルに名前を付けるときに使用する接頭部。この名前には、この名前が固有であることを保証するために、ObjectEventId が付加されます。

WaitForMDN

この属性は、文書の MDN を受信したという TPI からの通知をコネクターに待機させるかどうかを指定します。この属性はオプションです。この属性がメタオブジェクトに取り込まれると、コネクターの WaitForMDN プロパティーの指定は変更されます。デフォルト値は true です。

BackupRequired

この属性を指定すると、文書の送信後に TPI Server によって文書のバックアップを行うためのフラグが設定されます。この属性はオプションです。この属性がメタオブジェクトに取り込まれると、コネクターの BackupRequired プロパティーの指定内容が変更されます。この属性は、sendDocument() 呼び出しのパラメーターとして渡されます。デフォルト値は true です。

ビジネス・オブジェクトの構造に関するデータ・ハンドラー要件

TPI コネクターが使用する各データ・ハンドラーには、ビジネス・オブジェクトの構造に対する独自の要件があります。ビジネス・オブジェクトは、その変換のために呼び出されたデータ・ハンドラーの仕様に合致していることが必要です。これらの要件は、「*データ・ハンドラー・ガイド*」に記載されています。

子メタオブジェクトを持つサンプル・ビジネス・オブジェクト

以下に示す例は、基数が 1 の子としてのメタオブジェクトを持つ TPI コネクター・ビジネス・オブジェクトの定義です。このビジネス・オブジェクト定義は、区切り文字対応のデータ・ハンドラー向けに作成されています。

```
[BusinessObjectDefinition]
Name = TPICustBO
Version = 1.0.0.
AppSpecificInfo = cw_mo_cfg=CustBORouteInfo;

[Attribute]
Name = FirstName
Type = String
IsKey = true
IsRequired = true
AppSpecificInfo =
[end]

[Attribute]
Name = LastName
Type = String
IsKey = true
IsRequired = true
AppSpecificInfo =
[end]

[Attribute]
Name = Company
Type = String
IsKey = true
IsRequired = true
AppSpecificInfo =
[end]

[Attribute]
Name = City
Type = String
IsKey = false
IsRequired = false
AppSpecificInfo =
[end]

[Attribute]
Name = CustBORouteInfo
Type = TPIRouteInfo
ContainedObjectVersion = 1.0.0.
Relationship = Containment
```

```
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue =
AppSpecificInfo = type=cw_mo_cfg
[end]

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

[Verb]
Name = Create
[End]

[Verb]
Name = Retrieve
[End]

[End]

[BusinessObjectDefinition]
Name = TPIRouteInfo
Version = 1.0.0.

[Attribute]
Name = SenderId
Type = String
IsKey = true
IsRequired = true
AppSpecificInfo =
[end]

[Attribute]
Name = ReceiverId
Type = String
IsKey = true
IsRequired = true
AppSpecificInfo =
[end]

[Attribute]
Name = DocumentType
Type = String
IsKey = true
IsRequired = true
AppSpecificInfo =

[end]

[Attribute]
Name = BOPrefix
Type = String
IsKey = false
IsRequired = false
AppSpecificInfo =
[end]

[Attribute]
Name = WaitForMDN
```

```
Type = String
IsKey = false
IsRequired = false
AppSpecificInfo =
[end]

[Attribute]
Name = BackupRequired
Type = String
IsKey = false
IsRequired = false
AppSpecificInfo =
[end]

[Attribute]
Name = OriginalName
Type = String
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = TPICustomer
[End]

[Attribute]
Name = UniqueId
Type = String
IsKey = false
IsForeignKey = false
IsRequired = false
[end]

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

[Attribute]
Name = OriginalName
Type = String
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = TPICustomer
[End]

[Attribute]
Name = UniqueId
Type = String
IsKey = false
IsForeignKey = false
IsRequired = false
[end]

[Verb]
Name = Create
[End]

[Verb]
Name = Retrieve
[End]

[End]
```

マッピングの考慮事項 (WebSphere ICS 統合ブローカーのみ)

いくつかの TPI インプリメンテーション環境では、ある 1 つの企業に対して、ビジネス・オブジェクトは同じ (例えば、Sales_Order) でも、異なるフォーマットに基づいて文書を受信する取引先が複数存在する場合があります。このシナリオでは、ポリモアフィック・マップが要求されます。ポリモアフィック・マップでは、データを受信している取引先に応じてフォーマットの異なるビジネス・オブジェクトを出力できます。

ポリモアフィック・マップとは、本来は 2 つ以上の個別サブマップで、出力タイプごとに 1 つ存在します。同じ入力オブジェクトから、さまざまな種類の出力オブジェクトを生成します。これらのサブマップは 1 つのメイン・マップによって呼び出されます。メイン・マップは、属性値などの条件を検査して、呼び出すサブマップを決定します。サブマップが出力オブジェクトを生成すると、メイン・マップはこのオブジェクトをコネクターに返します。

この機能を TPI に対して実装する方法の 1 つは、ReceiverID 属性を判別条件として使用することです。メイン・マップは、入力オブジェクトの ReceiverID に応じて適切なサブマップを呼び出します。サブマップの作業を次に示します。

- 取引先が指定した文書フォーマットに応じて、適切なデータ・ハンドラーに対応する出力オブジェクトを生成する。
- 適切なメタデータを持つ子メタオブジェクトを取り込む。

ポリモアフィック・マップの詳細については、「*マップ開発ガイド*」を参照してください。

ビジネス・オブジェクト動詞の処理

TPI コネクターによって処理されたすべてのビジネス・オブジェクトには、必ず Create 動詞が設定されています。ある要求に対して Create 動詞が設定されていないと、コネクターは対象のビジネス・オブジェクトを不合格にします。Create 動詞がデータ・ハンドラーによって設定されていなかった場合、コネクターは、すべてのイベント・ビジネス・オブジェクトに対して Create 動詞を設定します。

ビジネス・オブジェクト属性プロパティー

TPI コネクターには、ビジネス・オブジェクト属性のプロパティーに対する特定の要件はありません。ビジネス・オブジェクト属性のプロパティーは、ビジネス・オブジェクトを変換するときに使用するデータ・ハンドラーの要件に適合する必要があります。これらの要件は、「*データ・ハンドラー・ガイド*」に記載されています。

ビジネス・オブジェクトのアプリケーション固有情報

ビジネス・オブジェクト定義でのアプリケーション固有情報には、ビジネス・オブジェクトの処理方法に関する指示をコネクターやデータ・ハンドラーに与える機能があります。TPI ビジネス・オブジェクトのアプリケーション固有情報は、対象のビジネス・オブジェクトを処理するために使用するコネクターとデータ・ハンドラーの両方の要件に適合する必要があります。

アプリケーション固有情報についてのコネクター要件

TPI コネクターは、アプリケーション固有情報をビジネス・オブジェクト・レベルで使用して、子メタオブジェクトを指定します。ビジネス・オブジェクトを変換する場合、WebSphere Business Integration Adapter に付属のデータ・ハンドラーでは、type = cw_mo_cfg タグが記述されている子メタオブジェクトの内容は、アプリケーション固有情報に対して、出力ストリームの一部としては組み込まれません。親オブジェクトのアプリケーション固有情報には、cw_mo_cfg タグを記述したメタオブジェクト名も指定しておく必要があります。

トップレベルのビジネス・オブジェクト・ヘッダーでは、アプリケーション固有情報に、メタオブジェクトの名前が次の構文で指定されます。

```
cw_mo_cfg=<meta-object attribute name>
```

メタオブジェクトの子ビジネス・オブジェクト属性では、アプリケーション固有情報により、メタオブジェクトの型が指定されます。このテキストが従う構文は次のとおりです。

```
type=cw_mo_cfg
```

アプリケーション固有情報についてのデータ・ハンドラー要件

アプリケーション固有情報についてのコネクター要件以外に、各ビジネス・オブジェクトを処理するために使用する特定のデータ・ハンドラーの要件について考慮する必要があります。各データ・ハンドラーのアプリケーション固有情報の要件の詳細については、「データ・ハンドラー・ガイド」を参照してください。

付録 A. コネクターの標準構成プロパティー

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

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

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

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

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

新規プロパティーと削除されたプロパティー

以下の標準プロパティーは、本リリースで追加されました。

新規プロパティー

- XMLNameSpaceFormat

削除されたプロパティー

- RestartCount

標準コネクター・プロパティーの構成

アダプター・コネクターには 2 つのタイプの構成プロパティーがあります。

- 標準構成プロパティー
- コネクター固有の構成プロパティー

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

Connector Configurator の使用

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

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

プロパティー値の設定と更新

プロパティー・フィールドのデフォルトの長さは 255 文字です。

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

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

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

- **動的**
  変更を System Manager に保管すると、変更が即時に有効になります。コネクターが System Manager から独立してスタンドアロン・モードで稼働している場合 (例えば、いずれかの WebSphere Message Brokers と連携している場合) は、構成ファイルでのみプロパティーを変更できます。この場合、動的更新は実行できません。
- コンポーネント再始動
  System Manager でコネクターを停止してから再始動しなければ、変更が有効になりません。アプリケーション固有コンポーネントまたは統合ブローカーを停止、再始動する必要はありません。
- サーバー再始動
  アプリケーション固有のコンポーネントおよび統合ブローカーを停止して再始動しなければ、変更が有効になりません。
- エージェント再始動 (ICS のみ)
  アプリケーション固有のコンポーネントを停止して再始動しなければ、変更が有効になりません。

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

標準プロパティーの要約

表 6 は、標準コネクター構成プロパティーの早見表です。標準プロパティーの依存関係は RepositoryDirectory に基づいているため、コネクターによっては使用されないプロパティーがあり、使用する統合ブローカーによってプロパティーの設定が異なる可能性があります。

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

表 6. 標準構成プロパティーの要約

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
AdminInQueue	有効な JMS キュー名	*CONNECTORNAME* /ADMININQUEUE	コンポーネント再始動	Delivery Transport は JMS
AdminOutQueue	有効な JMS キュー名	*CONNECTORNAME*/ADMINOUTQUEUE	コンポーネント再始動	Delivery Transport は JMS
AgentConnections	1 から 4	1	コンポーネント再始動	Delivery Transport は MQ および IDL: Repository Directory は <REMOTE>
AgentTraceLevel	0 から 5	0	動的	
ApplicationName	アプリケーション名	コネクター・アプリケーション名として指定された値	コンポーネント再始動	
BrokerType	ICS、WMQI、WAS			
CharacterEncoding	ascii7、ascii8、SJIS、Cp949、GBK、Big5、Cp297、Cp273、Cp280、Cp284、Cp037、Cp437 **注:** これは、サポートされる値の一部です。	ascii7	コンポーネント再始動	
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント再始動	Repository Directory は <REMOTE>
ContainerManagedEvents	値なしまたは JMS	値なし	コンポーネント再始動	Delivery Transport は JMS
ControllerStoreAndForwardMode	true または false	True	動的	Repository Directory は <REMOTE>
ControllerTraceLevel	0 から 5	0	動的	Repository Directory は <REMOTE>
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	コンポーネント再始動	JMS トランスポートのみ

表 6. 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
DeliveryTransport	MQ、IDL、または JMS	JMS	コンポーネント再始動	Repository Directory がローカルの場合は、値は JMS のみ
DuplicateEventElimination	True または False	False	コンポーネント再始動	JMS トランスポートのみ、Container Managed Events は <NONE> でなければならない
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory または CxCommon.Messaging.jms.SonicMQFactory または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	FactoryClassName が IBM の場合は crossworlds.queue.manager を使用。FactoryClassName が Sonicの場合 localhost:2506 を使用。	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMaxNativeStackSize	スタックのサイズ (キロバイト単位)	128k	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	Repository Directory は <REMOTE>
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならない

表 6. 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	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR **注:** これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	True または False	False	コンポーネント再始動	Repository Directory は <REMOTE> でなければならない
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならない
MessageFileName	パスまたはファイル名	InterchangeSystem.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	*CONNECTORNAME*/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は True でなければならない
OADAutoRestartAgent	True または False	False	動的	Repository Directory は <REMOTE> でなければならない
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければならない
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければならない
PollEndTime	HH:MM	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒) no (ポーリングを使用不可にする) key (コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力された場合にのみポーリングする)	10000	動的	

表 6. 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
PollQuantity	1 から 500	1	エージェント再始動	JMS トランスポートのみ: Container Managed Events を指定
PollStartTime	HH:MM (HH は 0 から 23、MM は 0 から 59)	HH:MM	コンポーネント再始動	
RepositoryDirectory	メタデータ・リポジトリーの場所		エージェント再始動	ICS の場合は <REMOTE> に設定する。WebSphere MQ Message Brokers および WAS の場合: C:¥crossworlds¥repository に設定する
RequestQueue	有効な JMS キュー名	*CONNECTORNAME*/REQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
ResponseQueue	有効な JMS キュー名	*CONNECTORNAME*/RESPONSEQUEUE	コンポーネント再始動	Delivery Transport が JMS の場合: Repository Directory が <REMOTE> の場合のみ必要
RestartRetryCount	0 から 99	3	動的	
RestartRetryInterval	適切な正数 (単位: 分): 1 から 2147483547	1	動的	
RHF2MessageDomain	mrm、xml	mrm	コンポーネント再始動	Delivery Transport が JMS であり、かつ WireFormat が CwXML である。
SourceQueue	有効な WebSphere MQ 名	*CONNECTORNAME*/SOURCEQUEUE	エージェント再始動	Delivery Transport が JMS であり、かつ Container Managed Events が指定されている場合のみ
SynchronousRequestQueue		*CONNECTORNAME*/SYNCHRONOUSREQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
SynchronousRequestTimeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	Delivery Transport は JMS
SynchronousResponseQueue		*CONNECTORNAME*/SYNCHRONOUSRESPONSEQUEUE	コンポーネント再始動	Delivery Transport は JMS

表 6. 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
WireFormat	CwXML、CwBO	CwXML	エージェント再始動	Repository Directory が <REMOTE> でない場合は CwXML。Repository Directory が <REMOTE> であれば CwBO
WsifSynchronousRequest Timeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	WAS のみ
XMLNameSpaceFormat	short、long	short	エージェント再始動	WebSphere MQ Message Brokers および WAS のみ

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

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

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

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

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

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

AgentTraceLevel

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

ApplicationName

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

BrokerType

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

CharacterEncoding

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

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

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

ConcurrentEventTriggeredFlows

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

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

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

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

- `Maximum number of concurrent events` プロパティーの値を増加して、コラボレーションが複数のスレッドを使用できるように構成します。
- 宛先アプリケーションのアプリケーション固有コンポーネントが複数の要求を並行して実行できることを確認します。つまり、このコンポーネントがマルチスレッド化されているか、またはコネクター・エージェント並列処理を使用でき、複

数プロセスに対応するよう構成されている必要があります。Parallel Process Degree 構成プロパティーに、1 より大きい値を設定します。

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

ContainerManagedEvents

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

デフォルト値は No value です。

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

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

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

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

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

ControllerStoreAndForwardMode

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

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

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

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

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

デフォルト値は true です。

ControllerTraceLevel

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

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

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

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

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

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

WebSphere MQ および IDL

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

- 非同期 (ASYNC) 通信:
  WebSphere MQ を使用すると、アプリケーション固有のコンポーネントは、サーバーが利用不能である場合でも、イベントをポーリングして永続的に格納することができます。
- サーバー・サイド・パフォーマンス:
  WebSphere MQ を使用すると、サーバー・サイドのパフォーマンスが向上します。最適化モードでは、WebSphere MQ はイベントへのポインターのみをリポジトリー・データベースに格納するので、実際のイベントは WebSphere MQ キュー内に残ります。これにより、サイズが大きい可能性のあるイベントをリポジトリー・データベースに書き込む必要がありません。

- エージェント・サイド・パフォーマンス:
  WebSphere MQ を使用すると、アプリケーション固有のコンポーネント側のパフォーマンスが向上します。WebSphere MQ を使用すると、コネクターのポーリング・スレッドは、イベントを選出した後、コネクターのキューにそのイベントを入れ、次のイベントを選出します。この方法は IDL よりも高速で、IDL の場合、コネクターのポーリング・スレッドは、イベントを選出した後、ネットワーク経由でサーバー・プロセスにアクセスしてそのイベントをリポジトリー・データベースに永続的に格納してから、次のイベントを選出する必要があります。

JMS

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

このプロパティーを `true` に設定すると、JMS 対応コネクターによるデリバリー・キューへの重複イベントのデリバリーが防止されます。この機能を使用するには、コネクターに対し、アプリケーション固有のコード内でビジネス・オブジェクトの

ObjectEventId 属性として一意のイベント ID が設定されている必要があります。これはコネクター開発時に設定されます。

このプロパティーは、false に設定することもできます。

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

FaultQueue

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

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

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

デフォルト値は 128M です。

JvmMaxNativeStackSize

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

デフォルト値は 128K です。

JvmMinHeapSize

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

デフォルト値は 1M です。

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

デフォルト値は crossworlds.queue.manager です。

jms.NumConcurrentRequests

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

デフォルト値は 10 です。

jms.Password

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

デフォルトはありません。

jms.UserName

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

デフォルトはありません。

ListenerConcurrency

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

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

Locale

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

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、以下のように説明されます。

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

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

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

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

LogAtInterchangeEnd

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

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

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

MaxEventCapacity

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

値は 1 から 2147483647 の間の正整数です。デフォルト値は 2147483647 です。

MessageFileName

コネクター・メッセージ・ファイルの名前です。メッセージ・ファイルの標準位置は ¥connectors¥messages です。メッセージ・ファイルが標準位置に格納されていない場合は、メッセージ・ファイル名を絶対パスで指定します。

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

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

MonitorQueue

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

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

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

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

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

デフォルト値は false です。

OADMaxNumRetry

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

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

デフォルト値は 1000 です。

OADRetryTimeInterval

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

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

デフォルト値は 10 です。

PollEndTime

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

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

PollFrequency

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

- ポーリング・アクション間のミリ秒数。

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

デフォルト値は 10000 です。

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

PollQuantity

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

PollStartTime

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

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

RequestQueue

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

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

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

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

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

ResponseQueue

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

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

RestartRetryCount

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

デフォルト値は 3 です。

RestartRetryInterval

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

デフォルト値は 1 です。

RHF2MessageDomain

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

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

サンプル・ヘッダーを以下に示します。

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

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

SourceQueue

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

JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、39 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

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

WsifSynchronousRequest Timeout

WAS 統合ブローカーでのみ使用されます。

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

デフォルト値は 0 です。

XMLNameSpaceFormat

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

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

デフォルト値は `short` です。

付録 B. Connector Configurator

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

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

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

注:

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

この付録では、次のトピックについて説明します。

Connector Configurator の概要

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

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

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

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

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

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

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

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

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

ただし**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクター固有プロパティーのテンプレートを作成します。システム内で既にテンプレートが作成されている場合には、作成されているテンプレートを使用します。システム内でまだテンプレートが作成されていない場合には、53 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

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

Connector Configurator の始動

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

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

スタンドアロン・モードでの Configurator の実行

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

これを行うには、以下のステップを実行します。

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

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

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (58 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

1. 「System Manager」ウィンドウの「コネクター」フォルダーでいずれかの構成ファイルを選択し、右クリックします。Connector Configurator が開き、この構成ファイルの統合ブローカー・タイプおよびファイル名が上部に表示されます。
2. 「標準のプロパティー」タブをクリックし、この構成ファイルに含まれているプロパティーを確認します。

コネクター固有のプロパティー・テンプレートの作成

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

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

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

新規テンプレートの作成

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

テンプレートは以下のように作成します。

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

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

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

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

一般特性の指定

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

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

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

値の指定

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

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

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

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

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

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

依存関係の設定

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

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

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

   == (等しい)

   != (等しくない)

> (より大)

< (より小)

>= (より大か等しい)

<= (より小か等しい)

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

新規構成ファイルの作成

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

ブローカーを選択するには、以下のステップを実行します。

- Connector Configurator のホーム・メニューで、**「ファイル」>「新規」>「コネクター構成」**をクリックします。**「新規コネクター」**ダイアログ・ボックスが表示されます。
- **「Integration Broker」**フィールドで、ICS 接続、WebSphere Message Brokers 接続、WAS 接続のいずれかを選択します。
- この章で後述する説明に従って**「新規コネクター」**ウィンドウの残りのフィールドに入力します。

また、以下の作業も実行できます。

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

コネクター固有のテンプレートからの構成ファイルの作成

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

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

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

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

- **システム接続**

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

- **「コネクター固有プロパティー・テンプレート」を選択します。**

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

  使用するテンプレートを選択し、**「OK」**をクリックします。

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

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

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

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

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

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

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

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

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

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

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

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

構成ファイルの完成

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

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

1. **「標準のプロパティー」**タブで、BrokerType プロパティーの値フィールドを選択します。ドロップダウン・メニューで、値 ICS、WMQI、または WAS を選択します。
2. 選択したブローカーに関連付けられているプロパティーが「標準のプロパティー」タブに表示されます。ここでファイルを保管するか、または 62 ページの『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。
3. 構成が完了したら、**「ファイル」>「保管」>「プロジェクトに」**を選択するか、または**「ファイル」>「保管」>「ファイルに」**を選択します。

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

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

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

構成ファイル・プロパティーの設定

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

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

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

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

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

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

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

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

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

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

- 背景がグレーのフィールドは、標準のプロパティーを表します。値を変更することはできますが、名前の変更およびプロパティーの除去はできません。
- 背景が白のフィールドは、アプリケーション固有のプロパティーを表します。これらのプロパティーは、アプリケーションまたはコネクターの特定のニーズによって異なります。値の変更も、これらのプロパティーの除去も可能です。
- 「値」フィールドは構成できます。
- **「更新メソッド」**フィールドは通知用であり、構成できません。このフィールドは、値が変更されたプロパティーをアクティブにするために必要なアクションを示します。

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下の手順を実行します。

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

を閉じると、すべてのカテゴリーで入力した値を一括して保管するかまたは破棄するかを確認するプロンプトが出されます。

- 修正した値を保管するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「はい」**をクリックします。「ファイル」メニューまたはツールバーから**「保管」>「ファイルに」**をクリックする方法もあります。

アプリケーション固有の構成プロパティーの設定

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

1. グリッドの左上端の部分で右マウス・ボタンをクリックします。ポップアップ・メニュー・バーが表示されます。プロパティーを追加するときは**「追加」**をクリックします。子プロパティーを追加するには、親の行番号で右マウス・ボタンをクリックし、**「子を追加」**をクリックします。
2. プロパティーまたは子プロパティーの値を入力します。
3. プロパティーを暗号化するには、**「暗号化」**ボックスを選択します。
4. 60 ページの『標準コネクター・プロパティーの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

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

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

コネクター・プロパティーの暗号化

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

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

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

更新メソッド

付録 A 『*コネクターの標準構成プロパティー*』の 32 ページの『プロパティー値の設定と更新』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

コネクターで使用するビジネス・オブジェクトを指定するには、Connector Configurator の**「サポートされているビジネス・オブジェクト」**タブを使用します。汎用ビジネス・オブジェクトと、アプリケーション固有のビジネス・オブジェクトの両方を指定する必要があり、またそれらのビジネス・オブジェクト間のマップの関連を指定することが必要です。

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

ご使用のブローカーが ICS の場合

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

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

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

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

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

サポートされるリストからビジネス・オブジェクトを削除すると、コネクター定義が変更され、削除されたビジネス・オブジェクトはコネクターのこのインプリメン

テーションで使用不可になります。コネクターのコードに影響したり、そのビジネス・オブジェクト定義そのものが System Manager から削除されることはありません。

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

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

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

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

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

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

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

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

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

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

ご使用のブローカーが WAS の場合

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

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

System Manager を実行している場合、「サポートされているビジネス・オブジェクト」タブの「ビジネス・オブジェクト名」列の下にある空のボックスを選択できま

す。コンボ・ボックスが表示され、コネクターが属する統合コンポーネント・ライブラリー・プロジェクトから選択可能なビジネス・オブジェクトのリストが示されます。このリストから必要なビジネス・オブジェクトを選択します。

関係付けられたマップ (ICS のみ)

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

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

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

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

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的**

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

  明示的バインディングが必要なのは、特定のサポートされるビジネス・オブジェクトに複数のマップが存在する場合のみです。ICS は、ブート時、各コネクターでサポートされるそれぞれのビジネス・オブジェクトにマップを自動的にバイン

ドしようとします。複数のマップでその入力データとして同一のビジネス・オブジェクトが使用されている場合、サーバーは、他のマップのスーパーセットである 1 つのマップを見付けて、バインドしようとします。

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

以下の手順を実行して、マップを明示的にバインドします。

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

リソース (ICS)

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

メッセージング (ICS)

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

トレース/ログ・ファイル値の設定

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

ログとトレースの値を変更するには、以下の手順を実行します。

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

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

- ファイルに:
  ログ・メッセージまたはトレース・メッセージを指定されたファイルに書き込みます。ファイルを指定するには、ディレクトリー・ボタン (省略符号) をクリックし、指定する格納場所に移動し、ファイル名を指定し、**「保管」**をクリックします。ログ・メッセージまたはトレース・メッセージは、指定した場所の指定したファイルに書き込まれます。

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

データ・ハンドラー

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

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

構成ファイルの保管

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

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

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

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

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

構成ファイルの変更

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

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

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

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

構成の完了

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

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

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

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

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

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

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

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

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

付録 C. IBM WebSphere DataHandler for RNIF over TPI

IBM WebSphere Business Integration Data Handler for RosettaNet Implementation Framework (RNIF) over TPI (*TPI-RNIF データ・ハンドラー*) を使用すると、Trading Partner Interchange (TPI) Server の使用時に取引先との RosettaNet 文書の交換が容易になります。データ・ハンドラーは、 TPI Server の要件に固有の XML メッセージを作成し、構文解析します。

この付録では、読者が Trading Partner Interchange (TPI) 製品、RosettaNet Implementation Framework (RNIF)、IBM WebSphere Adapter for TPI、および IBM WebSphere Business Integration Data Handler for XML に関連する基本的な用語と基本機能について熟知していることを前提としています。これらの製品および標準について詳しくは、以下を参照してください。

- TPI Server に付属の「*Administrator's Guide*」
- http://www.rosettanet.org
- 本書「*Adapter for Trading Partner Interchange ユーザーズ・ガイド*」のこれまでの章
- IBM WebSphere Business Integration Adapters の「*データ・ハンドラー・ガイド*」

この付録で提供するいくつかの例では、Chemical Industry Data Exchange (CDIX) 形式を使用しています。この形式は、 IBM WebSphere Adapter for TPI で広く実装されている RosettaNet の特殊化した使用法です。

概要

TPI Server と IBM WebSphere Adapter for TPI 間のメッセージの交換時に、TPI Server は、関係する取引先と交換する RosettaNet データの両方の記述に Message Control Document (MCD) を使用します。MCD は、TPI Server が RosettaNet 文書の処理に使用する TPI 固有の XML 文書です。 TPI Server は、MCD 内の情報を取引先との間で送受信する RosettaNet 文書に変換します。

TPI-RNIF データ・ハンドラーは、 MCD の作成と構文解析のプロセスを単純化します。また、MCD に組み込まれているさまざまな RNIF ヘッダー、ビジネス・シグナルおよび Partner Interface Processes (PIP) メッセージがあればこれらを認識し、処理します。このハンドラーは、通常、複数の PIP を使用する場合に、管理する必要のあるビジネス・オブジェクト定義の数を最小限に抑えます。

TPI-RNIF データ・ハンドラーは、 MCD をマップする必要のある適切なビジネス・オブジェクト定義を判別し、MCD に組み込まれている RosettaNet ペイロードの各種 RNIF パーツ、ビジネス・シグナル、および PIP 文書を識別します。TPI-RNIF データ・ハンドラーは、これらのパーツを識別すると、内部的に IBM WebSphere Business Integration Data Handler for XML を呼び出し、続いて IBM WebSphere Business Integration Data Handler for XML がこれらのパーツをそれぞれ該当するビジネス・オブジェクト・タイプに変換します。したがって、 TPI-RNIF データ・ハンドラーで使用されるビジネス・オブジェクト定義は、XML データ・ハ

ンドラーのビジネス・オブジェクト要件と、この付録で説明する TPI-RNIF データ・ハンドラーの命名規則の両方に準拠していなければなりません。

ソフトウェア前提条件

TPI-RNIF データ・ハンドラーを使用するためには、以下のソフトウェアをインストールしておく必要があります。

- IBM WebSphere Adapter for Trading Partner Interchange、v 3.4.0 以降
- IBM WebSphere Data Handler for XML、v 2.5.0 以降
- Trading Partner Interchange、v. 4.1.0 以降、MCD v. 2.0 のサポート付き

TPI-RNIF データ・ハンドラーを作成するためには、WebSphere Business Integration Adapter Framework ライブラリーと WebSphere Business Integration Data Handler for XML のクラス・ファイルの両方が必要です。

実行時に、WebSphere Data Handler for XML が存在している必要があります。

インストール

TPI Server をインストールする必要があります。 RosettaNet の各取引先のインバウンド・プロトコルまたはアウトバウンド・プロトコルとして RosettaNet がリストされていることを確認してください。また、取引先会社が、サポートされる XML 文書タイプとして RossettaNet および/または MCD をリストしていることを確認してください。TPI Server のインストールおよび構成の手順については、TPI Server に付属の「*Administrator's Guide*」を参照してください。

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

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

TPI-RNIF データ・ハンドラーのインストール・ファイルは、アダプター・システムの DataHandlers ディレクトリーにインストールされる、CwDataHandler.jar の一部として組み込まれます。インストール・ディレクトリーの構造を以下に示します。

¥DataHandlersw¥CwDataHandler.jar

アダプターのインストール時に、必ず以下の作業を完了してください。

1. アダプターを、TPI-RNIF データ・ハンドラーの構成情報を格納するデータ・ハンドラー構成メタオブジェクトを使用するように構成します。

   これを行うには、インストール時に組み込まれるサンプル・ビジネス・オブジェクト

   ```
   MO_DataHandler_CIDX
   ```

   にアダプターをサブスクライブします。その後、コネクター固有のプロパティー

   ```
   DataHandlerDefaultMO=MO_DataHandler_CIDX
   ```

を設定します。

2. TPI-RNIF データ・ハンドラーをご使用のデータ・ハンドラーのメタオブジェクト内で構成します。

   例えば、

   `MO_DataHandler_CIDX`

   をご使用の場合は、`MO_DataHandler_ChemXMLConfig` がご使用のビジネス・オブジェクト定義とぴったり一致する `TopBOPrefix`、`BOPrefix` などの値を持つことを確認してください。 9 ページの『取引先の構成ファイルの作成』を参照してください。

3. 送信側取引先と受信側取引先の両方で TPI-RNIF データ・ハンドラーを使用するように、アダプターを構成します。

   これを行うには、アダプターに指定する取引先構成ファイルを使用します。例えば、

   `MO_DataHandler_CIDX`

   を使用する場合、このファイルで、データを交換する取引先に MIME タイプの「text/tpi_rnif」を指定します。

4. アダプターに送信される各要求に組み込まれる TPI 子メタオブジェクトに、属性

   `DocumentType=CW_RNIF`

   を設定します。

   この属性を設定しないと、TPI Server が文書を MCD として認識せず、宛先となる取引先にロー XML として送信します。 23 ページの『子メタオブジェクトの属性』を参照してください。

TPI RNIF データ・ハンドラーの構成

TPI-RNIF データ・ハンドラーの構成は、データ・ハンドラー・メタオブジェクトを使用して行います。以下の表に示す必須プロパティーは、TPI-RNIF データ・ハンドラーに固有のものです。TPI-RNIF データ・ハンドラーは、既存の XML データ・ハンドラーの機能を内部的に拡張するため、 XML データ・ハンドラーに必要なこの他のメタオブジェクト・プロパティーについて詳しくは、「データ・ハンドラー・*ガイド*」を参照してください。

表 7. TPI-RNIF データ・ハンドラーの必須メタオブジェクト・プロパティー

プロパティー	説明	必須	デフォルト値

Top BO Prefix	特定のサービス・コンテンツ文書用のラッパー・ビジネス・オブジェクトを解決する。サンプル値には、
	`RNETOBJ`
	および
	`CIDXOBJ`
	がある。
		必須ではないが、指定を推奨。	なし

BO Prefix	TopBOPrefix と同様であるが、MCD に組み込まれている RNIF メッセージ・パーツ (プリアンブル、サービス・ヘッダーなど) をビジネス・オブジェクトに変換する。サンプル値には、
	`RNET`
	および
	`CIDX`
	がある。
		必須ではないが、指定を推奨。	なし

PIP Specific Signals	セミコロンで区切られた文書タイプのリスト。指定された文書タイプに対して、データ・ハンドラーがラッパー・ビジネス・オブジェクト名の解決時に PIP ID を組み込む。	いいえ	なし
Class 名前		はい	com.crossworlds.DataHandlers.text.tpi_rnif_mcd2

ビジネス・オブジェクトの構造について

TPI-RNIF データ・ハンドラーでは、処理するビジネス・オブジェクトのトップレベルの構造に特定の要件があります。このデータ・ハンドラーでは、トップレベルのビジネス・オブジェクトがラッパーとして機能し、ラッパー・オブジェクトが MCD および MCD 内に含まれる各種の RNIF パーツを表す複数の子オブジェクトを格納していることが要求されます。

ラッパー・オブジェクトの構造の例を次に示します。

```
Wrapper BO
      |MessageControlDocument BO
      |Preamble BO
      |ServiceHeader BO
      |ServiceContent BO
```

ラッパー・ビジネス・オブジェクトの命名

データ・ハンドラーでは、データ・ハンドラーがインバウンド MCD のビジネス・オブジェクト・インスタンスへの変換時に使用する適切なビジネス・オブジェクト定義を識別するために、特定の命名規則に従う必要があります。ラッパー・オブジェクトは、以下の規則に従って名前を付ける必要があります。

```
<TopBOPrefix_>Version_ServiceContentDoctype
```

ここで

```
TopBOPrefix
```

はオプションです。TPI-RNIF データ・ハンドラーは、

```
TopBOPrefix
```

の値を文書タイプの先頭に付加して、適切なラッパーを判別します。オブジェクト名を派生させる文書タイプは、常に MCD 内に含まれている RNIF サービス・コンテンツ・パーツの文書タイプ (例えば、OrderCreate、ReceiptAcknowledgement など) になります。

名前の

```
Version
```

パーツは、MCD のパッケージング・プロトコル記述に指定された値によって決まります。

パッケージング・プロトコル記述の例を次に示します。

```
<MessageControlDocument>
     <PackagingProtocol>
          <Standard>CIDX</Standard>
          <Version>2.0</Version>
     </PackagingProtocol>
</MessageControlDocument>
```

注: バージョン名にピリオドを指定した場合、ビジネス・オブジェクト名の解決時にピリオドが下線に置き換えられます。

ビジネス・オブジェクト名の解決例を次に示します。

1. TPI-RNIF データ・ハンドラーに

   ```
   TopBOPrefix=CIDXOBJ
   ```

   を設定します。
2. RNIF 2.0 で構造化された OrderCreate 要求が含まれた MCD を、データ・ハンドラーに受け渡します。

データ・ハンドラーは、MCD を

```
CIDXOBJ_2_0_OrderCreate
```

にマップします。

子ビジネス・オブジェクトの命名

データ・ハンドラーが子ビジネス・オブジェクトを正しく処理できるように、子ビジネス・オブジェクトもまた、命名規則に準拠している必要があります。MCD エンベロープを表す子オブジェクトは、以下の構造と合致している必要があります。

```
BOPrefix_Doctype
```

MCD エンベロープを表す子オブジェクトの例を以下に示します。

```
CIDX_MessageControlDocument
```

RNIF メッセージ・パーツ (プリアンブル、サービス・ヘッダー、およびサービス・コンテンツを含む) を表す子オブジェクトも正しく命名し、以下の構造と合致している必要があります。

```
BOPrefix_Version_Doctype
```

ここで

```
DocType
```

は、RNIF メッセージ・パーツを表します。RNIF メッセージのパーツを表す子オブジェクトの例を以下に示します。

```
CIDX_2_0_Preamble
```

子オブジェクトの属性の定義

データ・ハンドラーでは、ラッパーに少なくとも次の 4 つの子オブジェクト属性を定義する必要があります。

- `MessageControlDocument`
- `Preamble`
- `ServiceHeader`
- `ServiceContent`

以下の例は、RNIF 2.0 の仕様に従った RosettaNet ビジネス・オブジェクト定義に対応する必須属性を示しています。

```
RNETOBJ_2_0_OrderCreate
```

がラッパー・オブジェクトです。

```
TopBOPrefix=RNETOBJ
```

```
BOPrefix=RNET
```

表 8. タイプに対応する必須属性

属性名	属性のタイプ
MessageControlDocument	RNET_MessageControlDocument
Preamble	RNET_2_0_Preamble
ServiceHeader	RNET_2_0_ServiceHeader
ServiceContent	RRNET_2_0_OrderCreate

注: 属性を定義するときは、例に示されていないフィールドのデフォルト値を変更しないようにしてください。このようなフィールドには、カーディナリティー、アプリケーション固有の情報などが含まれています。

PIP 固有のシグナルの使用

トップレベルのオブジェクト・ラッパーに PIP ID を組み込むように、データ・ハンドラーを構成することができます。この指定はオプションですが、 RosettaNet シグナル・メッセージの交換時に効果的です。PIP 固有のシグナルを組み込む場合は、データ・ハンドラー・メタオブジェクト内で

```
PIPSpecificSignals
```

オプションを選択してから、すべての文書タイプを指定する必要があります。文書タイプはセミコロンで区切り、スペースなしで指定してください。データ・ハンドラーは、MCD 文書の受信時に、サービス・コンテンツの文書タイプが PIPSpecificSignals オプションに指定したタイプであるかどうか調べます。一致した場合、データ・ハンドラーは、MCD から PIP ID を抽出し、以下の構文を基にラッパー・オブジェクト名をシークします。

```
<TopBOPrefix_>Version_PIPType_ServiceContentDoctype
```

RosettaNet シグナル・メッセージの交換時、データ・ハンドラーが組み込まれた PIP ID を使用する構成になっていれば、データ・ハンドラーがシグナルが生成された PIP によって、一般シグナル・メッセージを識別可能なビジネス・オブジェクトに変換することができます。これは、プロセスが対象のシグナルのみにサブスクライブできることを意味します。

PIP 固有のシグナルがなければ、データ・ハンドラーは、以下に示すように、受取確認通知シグナルを一般ラッパー・オブジェクトに変換します。

```
CIDXOBJ_1_1_ReceiptAcknowledgement
```

PIP 固有のシグナルがある場合、データ・ハンドラーは、以下に示すように、受取確認通知シグナルを特定のラッパー・オブジェクトにマップすることができます。

```
CIDXOBJ_1_1_2A1_ReceiptAcknowledgement
```

以下に示す例は、RNIF 1.1 の仕様に従った RosettaNet ビジネス・オブジェクト定義に対応する必須属性を示しています。この例では、PIP 固有の ID を使用しています。この例では、

```
PIPSpecificSignals
```

に文書タイプ

```
OrderCreate
```

が含まれていることが前提となっています。

```
RNETOBJ_1_1_E41_OrderCreate
```

がラッパー・オブジェクトです。

```
TopBOPrefix=RNETOBJ
```

```
BOPrefix=RNET
```

表 9. PIP 固有の ID を使用する、タイプに対応する必須属性

属性名	属性のタイプ
MessageControlDocument	RNET_MessageControlDocument
Preamble	RNET_1_1_Preamble

表 9. PIP 固有の ID を使用する、タイプに対応する必須属性 (続き)

属性名	属性のタイプ
ServiceHeader	RNET_1_1_ServiceHeader
ServiceContent	RNET_1_1_E41_OrderCreate

注: RosettaNet の標準では、OrderCreate 文書は PIP E41 の分類に入ると明示しています。

デリバリー・ヘッダーの使用

ラッパー・オブジェクトには、5 番目の属性として、デリバリー・ヘッダーを定義することができます。これはオプションです。

要求の処理時に、データ・ハンドラーが既存の子属性を検出し、この名前でデータを取り込んだ場合、データ・ハンドラーは子オブジェクトを直列化し、アウトバウンド MCD にこれを追加します。

イベントを通知するとき、MCD がデリバリー・ヘッダー構造を持っている場合、データ・ハンドラーはラッパー・オブジェクトにデリバリー・ヘッダー属性が含まれているかどうか検査します。この属性が含まれていれば、データ・ハンドラーはこの属性にデリバリー・ヘッダー構造をマップします。

データ・ハンドラーは、ラッパー・オブジェクト内でデリバリー・ヘッダー属性を検出できなかった場合、警告を出します。ただし、MCD がデリバリー・ヘッダー構造を持っていない場合は、ラッパー・オブジェクトにデリバリー・ヘッダー属性が含まれていたとしても、データ・ハンドラーは警告を出しません。

以下の例は、RNIF 2.0 の仕様に従った RosettaNet ビジネス・オブジェクト定義に対応する必須属性を示しています。 このラッパー・オブジェクトは、PIP 固有のシグナルと DeliveryHeader 属性を使用します。

```
RNETOBJ_2_0_E41_OrderCreate
```

がラッパー・オブジェクトです。

```
TopBOPrefix=RNETOBJ
```

```
BOPrefix=RNET
```

表 10. デリバリー・ヘッダーを使用する、タイプに対応する必須属性

属性名	属性のタイプ
MessageControlDocument	RNET_MessageControlDocument
Preamble	RNET_2_0_Preamble
ServiceHeader	RNET_2_0_ServiceHeader
ServiceContent	RNET_2_0_E41_OrderCreate
DeliveryHeader	RNET_2_0_ServiceHeader

キー・リンク属性の使用

トップレベルのビジネス・オブジェクトに、意味のあるキーを組み込むと効果的です。これを行うには、TPI-RNIF データ・ハンドラーで、キー・リンクを使用しま

す。キー・リンクは、ラッパーでのみ定義され、サポートされます。キー・リンクは、子ビジネス・オブジェクトの 1 つの中にある重要な情報をラッパー・オブジェクトへコピーするように、データ・ハンドラーに指示します。

データ・ハンドラーは、MCD を受信し、これに対応するラッパー・オブジェクトを構成するときに、ラッパー内で検出したキー・リンク属性に、子オブジェクト内のネストされた属性を取り込みます。

キー・リンクを定義するには、次の手順を実行します。

1. ラッパー・ビジネス・オブジェクト内に属性を作成します。

   すでに説明した 4 つの必須属性用に予約されている名前以外の、任意の名前を指定できます。
2. 以下の構文に従って、アプリケーション固有のテキストを指定します。

   ```
   type=key;ref=someInnerAttribute
   ```

   ここで

   ```
   someInnerAttribute
   ```

   は、ラッパー・オブジェクトを基準とした、属性へのパスを定義します。

以下に、TPI-RNIF データ・ハンドラーによるキー・リンクの処理例を示します。

キー・リンク属性名:

```
InstanceId
```

アプリケーション固有の情報:

```
type=key;ref=MessageControlDocument.TLO.ProcessInfo.PIPInstanceID
```

データ・ハンドラーは、子属性の値

```
PIPInstanceID
```

をラッパー属性

```
InstanceId
```

にコピーすることにより、これを解決します。

ビジネス・オブジェクト定義の作成

TPI-RNIF データ・ハンドラーと互換性のあるビジネス・オブジェクト構造を作成するためには、3 つのタイプのビジネス・オブジェクト定義を作成する必要があります。

- MCD エンベロープ用のビジネス・オブジェクト定義の作成

  MCD ビジネス・オブジェクト定義が 1 つだけ必要です。 XML Object Disvovery Agent (ODA) を使用してこれを作成し、入力として TPI Server システムの MCD XSD を使用します。 XML ODA に指定したビジネス・オブジェクトのプレフィックスが TPI-RNIF データ・ハンドラーのメタオブジェクトに指定し

たプレフィックスと同一であるようにしてください。「*データ・ハンドラー・ガイド*」の『ビジネス・オブジェクト定義を作成するための XML ODA の使用』のセクションを参照してください。

ほとんどの MCD ビジネス・オブジェクト定義では、Business Object Designer ツールを使用したもう 1 つの手順をさらに実行する必要があります。このツールを使用して、以下の属性のアプリケーション固有の情報を変更し、

```
MessageControlDocument.ROOT.ManifestInfo.MessageContentInfo.Body.Body
```

「cdata」タイプにします。例えば、次のようになります。

```
mcd:Body;type=pcdata;notag
```

を、以下のように変更します。

```
mcd:Body;type=cdata;notag
```

- RNIF ヘッダー、ビジネス・シグナルおよびメッセージ用のビジネス・オブジェクト定義の作成

  交換を予定しているサービス・コンテンツ・メッセージのタイプごとに 1 つずつ、ビジネス・オブジェクト定義が必要です。また、サポートを予定している RNIF の各バージョンごとに 1 つずつ、プリアンブルおよびサービス・ヘッダー・ビジネス・オブジェクトが必要です。 BOPrefix に、RNIF バージョン番号を以下のように組み込みます。

  ```
  RNET_2_0
  ```

  これらのビジネス・オブジェクト定義も、XML ODA を使用して作成します。

- Business Object Designer ツールを使用した、ラッパー・ビジネス・オブジェクト定義の作成

  すでに説明した命名規則に従って、ラッパー・オブジェクトに名前を付け、必要に応じて、MCD エンベロープ・ビジネス・オブジェクト、該当する RNIF ヘッダー・ビジネス・オブジェクト、およびシグナルまたは PIP 固有のビジネス・オブジェクトを追加します。

  さらに、子メタオブジェクトを定義して、Adapter for TPI に必要な経路指定情報を指定する必要があります。 23 ページの『子メタオブジェクトの属性』を参照してください。

付録 D. Adapter for Trading Partner Interchange のサンプル・シナリオ

この付録では、 Adapter for Trading Partner InterChange (TPI) を使用した、取引先との間での XML 文書の交換について理解するためのサンプル・シナリオを記載します。ここに記載するシナリオでは、Adapter for TPI を使用して統合ソリューションを実装する 2 つの方法を扱っています。シナリオは、実際のインストール・システムや実在する商取引を反映しているわけではありません。また、この他のソリューションも考えられます。シナリオでは、以下の 2 つの実装を取り扱います。

1. IBM WebSphere InterChange Server での Adapter for Trading Partner Interchange の使用
2. IBM WebSphere MQ Integrator Broker での Adapter for Trading Partner Interchange の使用

インストール前の注意事項および前提事項

1. ご使用の IBM 接続および統合ソフトウェアの知識、さらには、TPI Server についての知識が必要です。
2. 使用可能な 2 台のコンピューター (物理的なマシン) が必要です。
3. 1 台のコンピューターに、ご使用予定の IBM 接続および統合ソフトウェアがインストール済みである必要があります。

   この付録のシナリオでは、このコンピューターを「IBM」と呼びます。

   Websphere InterChange Server でのインストールの場合:

   - WebSphere InterChange Server 4.x
   - WebSphere Adapter for TPI
   - TPI Server
   - WebSphere MQ

   IBM WebSphere MQ Integrator Broker でのインストールの場合:

   - WebSphere Business Integration Adapters 4.x
   - WebSphere Adapter for TPI
   - TPI Server
   - WebSphere MQ Integrator Broker
4. 2 台目のコンピューターに、TPI Server がインストール済みであることが必要です。

   この付録のシナリオでは、このコンピューターを「TPI」と呼びます。

注: この文書内で %CROSSWORLDS% とは、以下のいずれかを示します。

- **WebSphere MQ Integrator Broker をご使用の場合:** インストールされている WebSphere Business Integratation Adapters が格納されているフォルダーを指します。
- **Websphere InterChange Server をご使用の場合:** インストールされている IBM Websphere Integration Server 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 の形式で記述されます。AIX または Solaris で実行する場合は、適宜変更してください (例、%CROSSWORLDS%¥connectors であれば ${CROSSWORLDS}/connectors となります)。

Websphere InterChange Server の使用

この想定シナリオでは、 **IBM Corporation (IBM)** と **IBM Trading Partner (IBMTP)** という 2 つの取引先会社があり、注文情報を相互に交換します。 **IBM** は、「Port」というバックエンドの想定アプリケーションとの間の統合に、IBM WebSphere Adapter for TPI と IBM Websphere InterChange Server を使用しています。このアプリケーションは、PortConnector を使用します。

データ・フローの要約

repos_copy ファイルには、2 つのコラボレーション・オブジェクトが含まれており、それぞれが 1 方向の統合を処理する設計になっています。

IBM から IBMTP へ

1 つのコラボレーション・オブジェクトは、Port_To_TPI という名前の単純なパススルー・オブジェクトです。このコラボレーション・オブジェクトは、以下のデータ・フローに従って処理されます。

- PortConnector から TPI_Order オブジェクトを受信する。
- TPI_Order オブジェクトを Adapter for TPI に送信する。
- Adapter for TPI が TPI_Order を受信し、以下の処理を行う。
  1. XML DataHandler を使用して、これを XML 文書に変換する。
  2. XML 文書を HTTP に組み込む。
  3. XML 文書を組み込んだ HTTP を、取引先サイト **IBMTP** で稼働する TPI Server に送信する。

IBMTP から IBM へ

2 つ目のコラボレーション・オブジェクトは、TPI_To_Port という名前のオブジェクトで、以下のデータ・フローで使用されます。

- **IBMTP** は、注文の XML 文書を格納する組み込み HTTP を **IBM** にある Adapter for TPI に送信する。
- Adapter for TPI が、組み込み HTTP を受信し、以下の処理を行う。
  1. XML データ・ハンドラーを使用して、XML 文書をビジネス・オブジェクトに変換する。
  2. このビジネス・オブジェクトを 2 つ目のコラボレーション・オブジェクトである TPI_To_Port に送信する。

- TPI_To_Port は、PortConnector を使用して、このビジネス・オブジェクトを Port アプリケーションに送信する。

InterChange Server でのサンプル・シナリオのインストール

1. **IBM Corporation** の TPI Server をセットアップします。以下に手順を示します。
   a. **IBM** のマシンで、TPI Administrator Tool を始動します。
   b. 「会社プロファイル（Company Profiles)」タブから、ファイル %CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereICS¥IBM¥IBMCorporation_company.xml からの会社プロファイルのインポートを行います。
   c. インポート・パスワードは空白のままにします。
   d. Administrator Tool から、パートナー・タブを選択し、ファイル %CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereICS¥IBM¥IBMTradingPartner_partner.xml からパートナー・プロファイルをインポートします。

      この TPI Server の会社 ID は、**IBM** になります。
   e. パートナー・プロファイルのプロパティーを開き、「アウトバウンド・トランスポート (Outbound Transport)」タブから、組み込み HTTP までナビゲートします。URL 「http://IBMHost:4080/exchange/IBM」をマシン名に合わせて変更します。
   f. TPHost を IBM Trading Partner のマシン名に更新します。
2. **IBM Trading Partner** の TPI Server をセットアップします。以下に手順を示します。
   a. **IBMTP** のマシンで、TPI Administrator Tool を始動します。
   b. 「会社プロファイル (Company Profiles)」タブから、ファイル %CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereICS¥IBMTP¥IBMTradingPartner_company.xml からの会社プロファイルのインポートを行います。
   c. Administrator Tool 内から、パートナー・タブを選択し、ファイル %CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereICS¥IBMTP¥IBMCorporation_partner.xml からパートナー・プロファイルをインポートします。

      この TPI Server の会社 ID は、**IBMTP** になります。
   d. パートナー・プロファイルのプロパティーを開き、「アウトバウンド・トランスポート (Outbound Transport)」タブから、組み込み HTTP までナビゲートします。URL 「http://IBMHost:4080/exchange/IBM」をマシン名に合わせて変更します。
   e. IBMHost を IBM のマシン名に更新します。
3. ビジネス・オブジェクトをリポジトリーへロードします。
   a. IBM マシンから WebSphere InterChange Server を始動します。
   b. Business Object Designer を使用して、リポジトリー・ファイル

Sample_TPI_Order_Objects.in

を

%CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereICS¥IBM

フォルダーからロードします。

c. ビジネス・オブジェクト 10 個がロードされたことを確認してください。

4. コネクターをリポジトリーへロードします。

a. Connector Configurator を使用して、リポジトリー・ファイル

Sample_TPI_Order_Connectors.in

を

%CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereICS¥IBM

フォルダーからロードします。

b. TPI Connector および PortConnector の定義がロードされたことを確認してください。

5. TPI Connector を構成します。

a. System Manager を使用して、TPI Adapter の定義を選択し、 Connector Designer を起動します。ご使用のファイル構造に合わせて、以下のアダプター構成プロパティーを変更します。これらが存在しない場合は、これらのパスおよびファイルを作成してください。

- TradingPartnerConfigurationFile
- MetaEventDir
- DocumentOutDir
- ArchiveProcessedDocDir

6. コラボレーション・オブジェクトおよびテンプレートをリポジトリーへロードします。

a. System Manager を使用して、

%CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereICS¥IBM

フォルダー内にある

Sample_TPI_Order_Collaborations.in

という名前のリポジトリー・ファイルをロードします。

b. 以下のように、

Order_PassThrough

テンプレート定義と、

Port_To_TPI

および

TPI_To_Port

のコラボレーション・オブジェクトがロードされたことを確認します。

7. コラボレーション・テンプレートをコンパイルします。 System Manager を使用して、 Collaboration Templates フォルダーを選択してから、「すべてコンパイル」を選択します。
8. WebSphere InterChange Server を再始動します。

サービス呼び出し要求シナリオの実行

サービス呼び出しシナリオを実行するには、以下の手順で行います。

1. **IBM マシン上で:**
   a. WebSphere InterChange Server を始動し、次に Adapter for TPI を始動します。(この操作により、自動的に TPI Server が始動します。)
   b. Visual Test Connector の 1 つのインスタンスを始動します。
2. **IBMTP マシン上で:** TPI Server を始動します。
3. ポート・コネクターをシミュレートします。
   a. Test Connector を使用して、PortConnector のプロファイルを定義します。
   b. Test Connector のファイル・メニューから、「エージェントの接続」を選択して、エージェントのシミュレートを開始します。
4. テスト・データをロードします。
   a. PortConnector をシミュレートする Visual Test Connector を使用して、「編集」メニューから「ビジネス・オブジェクトをロード」を選択します。
   b. 以下のファイルを選択し、ロードします。

      ```
      %CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereICS¥IBM¥sampleOrderData.bo
      ```
5. テスト・データを送信します。
   a. PortConnector をシミュレートする Visual Test Connector を使用して、ロードされたテスト・ビジネス・オブジェクトを選択します。
   b. 「要求」メニューから「送信」を選択します。
6. **IBMTP マシン** 上の XLMin フォルダー内に XML 文書があるかどうか調べることにより、処理が正常に終了したことを確認します。

ポーリング・シナリオの実行

ポーリング・シナリオを実行するには、以下の手順で行います。

1. **IBM マシン上で:**
   a. WebSphere InterChange Server を始動し、次に Adapter for TPI を始動します。(この操作により、自動的に TPI Server が始動します。)
   b. Visual Test Connector の 1 つのインスタンスを始動します。
2. **IBMTP マシン上で:** TPI Server を始動します。
3. Port Connector をシミュレートします。**IBM マシン上で:**
   a. Test Connector を使用して、PortConnector のプロファイルを定義します。
   b. Test Connector のファイル・メニューから、「エージェントの接続」を選択して、エージェントのシミュレートを開始します。
4. サンプル・データを送信します。
   a. **IBMTP マシン上で:** 以下のファイルを XMLout ディレクトリーへ移動します。

%CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereICS¥IBMTP¥sampleOrderData.xml

b. **IBMTP マシン** で稼働している TPI Server がこのイベントを処理し、 **IBM マシン** で稼働している Adapter for TPI に引き渡します。

5. PortConnector をシミュレートする Visual Test Connector を使用した要求を受け入れます。正常応答で応答します。
6. 受け入れ済みの要求のデータを検査して、処理が正常に終了したことを確認します。

WebSphere MQ Integrator Broker の使用

この想定シナリオでは、2 つの取引先会社 (「IBM Corporation」と「IBM Trading Partner」) が、注文情報を相互に交換します。 **IBM Corporation (IBM)** は、TPI Adapter を WebSphere MQ Integrator Broker との連携で使用して、自身と **IBM Trading Partner (IBMTP)** という名前の取引先との間の XML 文書の交換を行っています。

このサンプル・シナリオには、2 つの部分があります。各部分が、1 方向のメッセージ交換を処理する設計になっています。 1 つの部分では、 WebSphere MQ Integrator Broker が MQ Series キューにイベントを格納し、これを Adapter for TPI が読み取ります。このアダプターは、 XML DataHandler を使用して、メッセージを XML 文書に変換し、変換後の文書を **IBM Trading Partner** に送信します。

もう 1 つの部分では、**IBM Trading Partner** という会社が XML 文書を **IBM Corporation** に送信する必要があります。 **IBM Corporation** にある TPI Adapter が、 XML 文書を MQ Series メッセージに変換し、そのメッセージを MQ Series キューに格納します。これを、WebSphere MQ Integrator Broker が読み取ります。

注: このサンプルでは、実際の WebSphere MQ Integrator Broker は使用しません。TPI Adapter は、単に MQ Series キューを読み書きするのみです。Visual Test Connector を使用して、WebSphere MQ Integrator Broker をシミュレートします。

サンプル・シナリオのインストール

1. **IBM Corporation** の TPI Server をセットアップします。以下に手順を示します。
   a. **IBM** のマシンで、TPI Administrator Tool を始動します。
   b. 「会社プロファイル (Company Profiles)」タブから、ファイル %CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereMQIntegratorBroker ¥IBM¥IBMCorporation_company.xml からの会社プロファイルのインポートを行います。
   c. インポート・パスワードは空白のままにします。
   d. Administrator Tool から、パートナー・タブをクリックし、ファイル %CROSSWORLDS%¥connectors¥TPI¥samples¥ WebSphereMQIntegratorBroker¥IBM¥IBMTradingPartner_partner.xml からパートナー・プロファイルをインポートします。
   e. このTPI Server の会社 ID は、**IBM** になります。

2. **IBM Trading Partner** の TPI Server をセットアップします。以下に手順を示します。
   a. **IBMTP** のマシンで、TPI Administrator Tool を始動します。
   b. 「会社プロファイル (Company Profiles)」タブから、ファイル %CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereMQIntegratorBroker¥IBMTP¥IBMTradingPartner_company.xml からの会社プロファイルのインポートを行います。
   c. Administrator Tool から、パートナー・タブをクリックし、ファイル %CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereMQIntegratorBroker¥IBMTP¥IBMCorporation_partner.xml からパートナー・プロファイルをインポートします。
   d. この TPI Server の会社 ID は、**IBMTP** になります。
3. MQ Series をセットアップします。
   a. MQ Series キュー・マネージャーを作成して始動します。チャネル・イニシエーターとリスナーも実行します。
   b. 次の名前のキューを作成します。
      - ADMININQUEUE
      - ADMINOUTQUEUE
      - DELIVERYQUEUE
      - FAULTQUEUE
      - REQUESTQUEUE
4. %CROSSWORLDS%¥connectors¥TPI ディレクトリーにある start_TPI ファイルを開きます。CYCLONEHOMEDIR の値を TPI のホーム・ディレクトリーのパスに変更します。 ファイルを保存して閉じます。
5. TPI Adapter と Port Connector CFG ファイルを構成します。
   a. %CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereMQIntegratorBroker¥TPIConnector.cfg ファイルを開きます。
   b. 現在のセットアップに応じて、ファイル内の以下のプロパティーを変更します。詳細については、「IBM WebSphere Business Integration Adapter for TPI ユーザーズ・ガイド」を参照してください。プロパティーがパスまたはファイル名を表し、このパスまたはファイル名がまだ存在しない場合は、作成する必要があります。
      - QueueManager
      - RepositoryDirectory
      - TradingPartnerConfigurationFile
      - MetaEventDir
      - DocumentOutDir
      - ArchiveProcessedDocDir
   c. %CROSSWORLDS%¥connectors¥TPI¥samples¥WebSphereMQIntegratorBroker¥PortConnector.cfg ファイルを開きます。現在のセットアップに応じて、ファイル内の以下のプロパティーを変更します。プロパティーがパスまたはファイル名を表し、このパスまたはファイル名がまだ存在しない場合は、作成する必要があります。
      - QueueManager

- RepositoryDirectory

6. TPI Adapter CFG ファイルを指定します。
   - Windows の場合: TPI Adapter のショートカットのプロパティーを開きます。ターゲットの最後の引き数として、「-c」+ <TPIConnector.cfg ファイルの絶対パスおよびファイル名> を追加します。
     - 例: -c%CROSSWORLDS%¥connectors¥TPI¥Samples¥WebSphereMQIntegratorBroker¥IBM¥TPIConnector.cfg
   - UNIX の場合: ${CROSSWORLDS}/bin/connector_manager_TPI ファイルを開きます。AGENTCONFIG_FILE プロパティーの値として、「-c」+ <TPIConnector.cfg ファイルの絶対パスおよびファイル名> を設定します。
     - AGENTCONFIG_FILE=-c${CROSSWORLDS}/connectors/TPI/samples/WebSphereMQIntegratorBroker/IBM/TPIConnector.cfg

要求処理シナリオの実行

要求処理シナリオを実行するには、以下の手順で行います。

1. **IBM マシン上で:**
   a. Adapter for TPI を始動します。(この操作により、自動的に TPI Server が始動します。)
   b. Visual Test Connector の 1 つのインスタンスを始動します。
2. **IBMTP マシン上で:** TPI Server を始動します。
3. ポート・コネクターをシミュレートします。
   a. Visual Test Connector を使用して、PortConnector のプロファイルを定義します。
   b. Test Connector の「ファイル」メニューから「プロファイルを開く」を選択し、コネクター構成ファイルとして以下のファイルを指定します。

      ```
      %CROSSWORLDS%¥connectors¥TPI¥samples¥
      WebSphereMQIntegratorBroker¥IBM¥PortConnector.cfg
      ```

   c. エージェントを接続します。
4. テスト・データをロードします。
   a. PortConnector をシミュレートする Visual Test Connector を使用して、「編集」メニューから「ビジネス・オブジェクトをロード」を選択します。
   b. 以下のファイルを選択し、ロードします。

      ```
      %CROSSWORLDS%¥connectors¥TPI¥samples¥
      WebSphereMQIntegratorBroker¥IBM¥sampleOrderData.bo
      ```

5. テスト・データを送信します。
   a. PortConnector をシミュレートする Visual Test Connector を使用して、ロードされたテスト・ビジネス・オブジェクトを選択します。
   b. 「要求」メニューから「送信」を選択します。
6. **IBMTP マシン** 上の XLMin フォルダー内に XML 文書があるかどうか調べることにより、処理が正常に終了したことを確認します。

ポーリング・シナリオの実行

ポーリング・シナリオを実行するには、以下の手順で行います。

1. **IBM マシン上で:**
   a. Adapter for TPI を始動します。(この操作により、自動的に TPI Server が始動します。)
   b. Visual Test Connector の 1 つのインスタンスを始動します。
2. **IBMTP マシン上で:** TPI Server を始動します。
3. ポート・コネクターをシミュレートします。
   a. Visual Test Connector を使用して、PortConnector のプロファイルを定義します。
   b. Test Connector の「ファイル」メニューから「プロファイルを開く」を選択し、コネクター構成ファイルとして以下のファイルを指定します。

      ```
      %CROSSWORLDS%¥connectors¥TPI¥samples¥
      WebSphereMQIntegratorBroker¥IBM¥PortConnector.cfg
      ```

   c. エージェントを接続します。
4. サンプル・データを送信します。
   a. **IBMTP マシン上で:** 以下のファイルを XMLout ディレクトリーへ移動します。

      ```
      %CROSSWORLDS%¥connectors¥TPI¥samples¥
      WebSphereMQIntegratorBroker¥IBMTP¥sampleOrderData.xml
      ```

   b. **IBMTP マシン** で稼働している TPI Server がこのイベントを処理し、 **IBM マシン** で稼働している Adapter for TPI に引き渡します。
5. PortConnector をシミュレートする Visual Test Connector を使用した要求を受け入れます。正常応答で応答します。
6. 受け入れ済みの要求のデータを検査して、処理が正常に終了したことを確認します。

特記事項

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

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

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

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

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

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

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

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

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

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

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

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

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

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

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

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

プログラミング・インターフェース情報

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

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

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

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

商標

以下は、IBM Corporation の商標です。

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

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

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

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

WebSphere Business Integration Adapter Framework V2.4.0

Printed in Japan

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