WebSphere Business Integration Adapters

Adapter for SAP Exchange Infrastructure (SAP XI) ユーザーズ・ガイド

バージョン 1.0.x

WebSphere Business Integration Adapters

Adapter for SAP Exchange Infrastructure (SAP XI) ユーザーズ・ガイド

バージョン 1.0.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、97 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 2.4.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 SAP Exchange Infrastructure (SAP XI) User Guide V1.0.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 1 刷 2004.1

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

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

目次

本リリースの新機能

リリース 1.0.0 の新機能

SAP Exchange Infrastructure 用アダプターのバージョン 1.0.0

本書について

IBM(R) WebSphere(R) Business Integration Adapter ポートフォリオは、先進の e-business テクノロジーやエンタープライズ・アプリケーションに統合コネクティビティーを提供します。このシステムには、ビジネス・プロセス統合の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やテンプレートが含まれます。

本書では、SAP Exchange Infrastructure (SAP XI) 用アダプターの構成、およびビジネス・オブジェクト開発について説明します。

対象読者

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

本書の前提条件

本書の読者は、WebSphere Business Integration システム、ビジネス・オブジェクトとコラボレーションの開発、および SAP Exchange Interface 仕様について十分理解しているものとします。

関連文書

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

本書には、別の資料として、「*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)*」、および「*System Implementation Guide for WebSphere InterChange Server*」を参照する箇所が多数あります。本書を印刷する場合は、必要に応じてこれらの資料も印刷してください。

関連資料は、以下のサイトからインストールできます。

- 一般アダプター情報、WebSphere Message Broker (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) とともにアダプターを使用する場合、および WebSphere Application Server とともにアダプターを使用する場合:
  http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- InterChange Server とともにアダプターを使用する場合:
  http://www.ibm.com/websphere/integration/wicserver/infocenter
- メッセージ・ブローカー (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) の詳細情報:
  http://www.ibm.com/software/integration/mqfamily/library/manualsa/

- WebSphere Application Server の詳細情報:
  http://www.ibm.com/software/webservers/appserv/library.html

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

表記上の規則

本書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、記述されたとおりの値を示します。
太字	初出語を示します。
イタリック、***イタリック***	変数名または相互参照を示します。
青のアウトライン	マニュアルをオンラインで表示するときにのみ見られる青のアウトラインは、相互参照用のハイパーリンクです。アウトラインの内側をクリックすると、参照先オブジェクトにジャンプします。
`{ }`	構文の記述行の場合、中括弧で囲まれた部分は、選択対象のオプションです。1 つのオプションだけを選択する必要があります。
`[ ]`	構文の記述行の場合、大括弧 [ ] で囲まれた部分は、オプションのパラメーターです。
`...`	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、`option[,...]` 複数のオプションをコンマで区切って指定できることを意味します。
`< >`	命名規則により、1 つの名前の個々の要素を互いに区別するために、不等号括弧によって個々の要素が囲まれます。例えば、`<server_name><connector_name>tmp.log` のように使用します。
`/` おぴょび ¥	本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。UNIX システムの場合には、円記号をスラッシュ (/) に置き換えてください。すべての製品のパス名は、ご使用のシステムで製品がインストールされているディレクトリーを基準とした相対パス名です。
%text% および *$text*	% 記号で囲まれたテキストは、Windows の `text` システム変数またはユーザー変数の値を示します。UNIX 環境での同等の表記は *$text* です。これは、*text* UNIX 環境変数の値を示します。
ProductDir	製品のインストール先ディレクトリーを表します。

第 1 章 概要

SAP Exchange Infrastructure 用コネクターは、WebSphere Business Integration Adapter for SAP Exchange Infrastructure のランタイム・コンポーネントです。このコネクターによって WebSphere 統合ブローカーは、SAP Exchange Infrastructure メッセージを送受信するアプリケーションとビジネス・オブジェクトを交換できます。この章では、コネクター・コンポーネントおよび関連する Business Integration システム体系について説明します。

コネクターは、アプリケーション固有のコンポーネントとコネクター・フレームワークの 2 種類で構成されています。アプリケーション固有のコンポーネントには、特定のアプリケーションに合わせたコードが格納されています。コネクター・フレームワークのコードはすべてのコネクターに共通なので、コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントとの仲介役の機能を果たします。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントとの間で以下のようなサービスを提供します。

- ビジネス・オブジェクトの送信および受信
- 始動メッセージや管理メッセージの交換の管理

この資料には、アプリケーション固有のコンポーネントとコネクター・フレームワークに関する情報が記載されています。この資料では、この 2 つのコンポーネントのことを、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は、「*IBM WebSphere InterChange Server システム管理ガイド*」を参照してください。

注: すべての WebSphere Business Integration Adapter は、統合ブローカーとともに作動します。SAP Exchange Infrastructure 用コネクターは、以下の統合ブローカーとともに作動します。

- InterChange Server。この統合ブローカーの詳細は、「*テクニカル入門 (IBM WebSphere InterChange Server)*」を参照してください。
- WebSphere Application Server (WAS)。この統合ブローカーの詳細は、「*アダプター実装ガイド (WebSphere Application Server)*」を参照してください。

SAP Exchange Infrastructure 用アダプター 1.0.0 は、SAP Exchange Infrastructure 標準の主要セグメントをサポートすることを目的としています。SAP Exchange Infrastructure (SAP XI) は、システム間ビジネス・プロセスの実装を可能にします。

アダプターのアーキテクチャー

SAP XI アダプターを使用することにより、IBM WebSphere Business Integration Collaborations は、SAP XI との間でビジネス・オブジェクトを非同期で交換できます。アダプターは、Java™ Message Service (JMS) の MQ インプリメンテーションを使用します。JMS は、エンタープライズ・メッセージング・システムにアクセスするための API であり、これによって保証付きイベント・デリバリーも可能になります。

SAP XI 統合サーバーと SAP XI アダプターの間のメッセージ交換は、WebSphere MQ キューを使って行われます。SAP XI 側では、WebSphere MQ との対話用に JMS インバウンド・アダプターおよび JMS アウトバウンド・アダプターが構成されます。

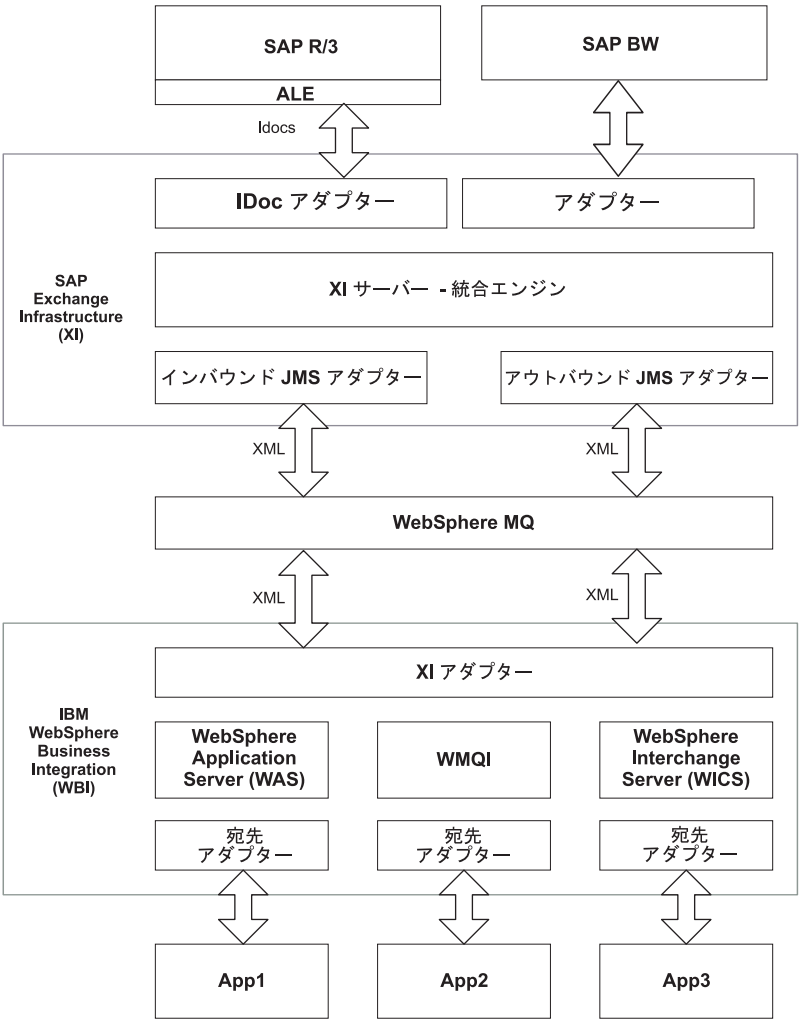

図 1. 体系図

SAP Exchange Infrastructure のアダプターはメタデータ主導型です。メッセージ・ルーティングとフォーマット変換がイベント・ポーリング技法で開始されます。イベント処理では、アダプターは、キューから SAP XI メッセージを検索し、データ・ハンドラーを呼び出してそれらのメッセージを対応するビジネス・オブジェクトに変換します。そして、そのビジネス・オブジェクトを統合ブローカーにデリバリーします。反対の方向となる要求処理では、アダプターは、統合ブローカーからビジネス・オブジェクトを受信し、同じデータ・ハンドラーを使用して SAP XI メッセージに変換します。そして、そのメッセージを SAP XI 統合サーバーにデリバリーします。

SAP XI アダプターは、XML データ・ハンドラーを使用して、SAP XI メッセージを処理します。このデータ・ハンドラーを SAP Exchange Infrastructure コネクターと連動するように構成する方法については、「*データ・ハンドラー・ガイド*」の第 3 章『XML データ・ハンドラー』を参照してください。

XML およびビジネス・オブジェクト

メッセージの処理に使用されるビジネス・オブジェクトのタイプおよび動詞は、メッセージ本文に含まれる XML ルート・エレメントの名前に基づきます。コネクターは、メタオブジェクトの項目を使用して、ビジネス・オブジェクト名と動詞を決定します。ユーザーは、各メッセージについて、ビジネス・オブジェクト名と動詞を保管するメタオブジェクトを構成し、SAP Exchange Infrastructure メッセージのルート・エレメントに関連付けます。

動的メタオブジェクトを構成し、それをコネクターに渡されるビジネス・オブジェクトに子として追加することができます。この子メタオブジェクトの値は、コネクター全体に指定された静的メタオブジェクトの値をオーバーライドします。子メタオブジェクトが定義されていない、または必要な変換プロパティーが定義されていない場合、コネクターはデフォルトで静的メタオブジェクトの値を検査します。1 つの静的コネクター・メタオブジェクトの代わり、または補足として、1 つ以上の動的子メタオブジェクトを指定できます。

コネクターは、複数の入力キューをポーリングし、各キューをラウンドロビン方式でポーリングして指定された数のメッセージを各キューから検索することができます。ポーリング中に検索された各メッセージに対して、コネクターは動的子メタオブジェクトを追加します (ビジネス・オブジェクトに指定されている場合)。子メタオブジェクトの値に応じて、コネクターは、メッセージ・フォーマットやメッセージが検索された入力キューの名前をビジネス・オブジェクトの属性に取り込みます。

入力キューからメッセージが検索されると、コネクターは、XML ルート・エレメント名を使用してビジネス・オブジェクト名を検索します。そしてメッセージ本文は、該当するビジネス・オブジェクトの新規インスタンスとともに、データ・ハンドラーに渡されます。このフォーマットに関連するビジネス・オブジェクト名が検出されない場合は、メッセージ本文のみがデータ・ハンドラーに渡されます。ビジネス・オブジェクトにメッセージ内容が正常に取り込まれると、コネクターは、そのビジネス・オブジェクトがサブスクライブされているかを確認してから、`gotApplEvents()` メソッドを使って統合ブローカーにデリバリーします。

アプリケーションとコネクターの通信

コネクターは、Java Message Service (JMS) のインプリメンテーションである IBM の WebSphere MQ を使用します。JMS は、エンタープライズ・メッセージング・システムにアクセスするためのオープン・スタンダード API です。これは、ビジネス・アプリケーションがビジネス・データとイベントを非同期で送受信できるように設計されています。

メッセージ要求

図 2 に、メッセージ要求通信を示します。`doVerbFor()` メソッドが統合ブローカーから WebSphere Business Integration システムのビジネス・オブジェクトを受信すると、コネクターは、そのビジネス・オブジェクトをデータ・ハンドラーに渡します。データ・ハンドラーは、ビジネス・オブジェクトを XML に変換し、コネクターはそれをメッセージとしてキューに発行します。ここで JMS 層は、適切な呼び出しを実行してキュー・セッションを開き、メッセージの経路を指定します。

SAP XI をサポートするインターフェースごとに、異なるキューにバインドされた別々のインバウンド JMS アダプター・インスタンスの定義が必要になります。例えば、MATMAS と ORDERS の IDoc の要求処理をサポートしたい場合は、異なる 2 つのインバウンド JMS アダプター・インスタンスを SAP XI に構成する必要があります。WebSphere Adapter for SAP Exchange Infrastructure は、同じインスタンスで複数の要求メッセージを処理し、異なるキューに POST することができます。

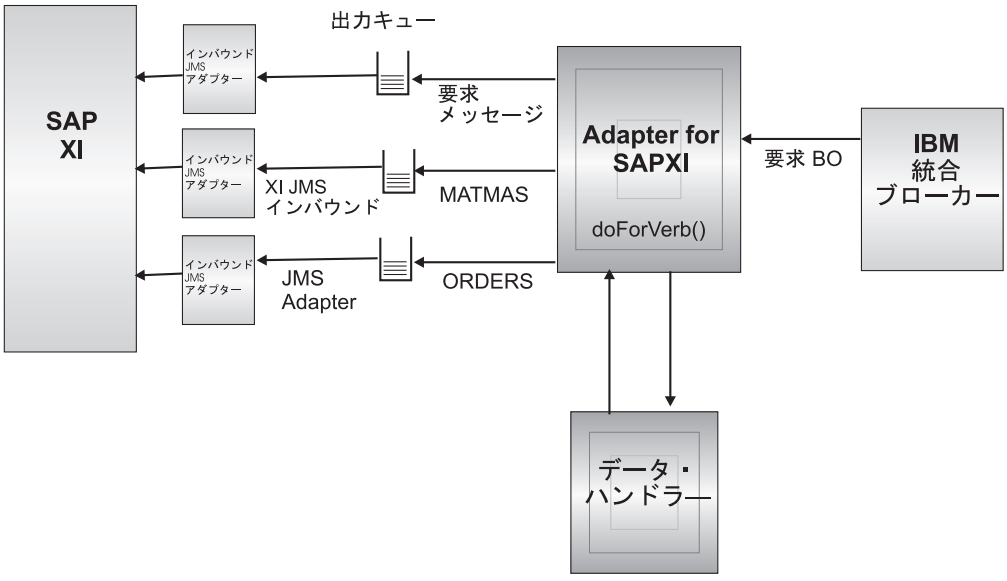

図 2. アプリケーションとコネクターの通信方式: メッセージ要求

イベント・デリバリー

図 3 に、イベント・デリバリーの方向を示します。SAP XI アウトバウンド JMS アダプターが WebSphere Adapter for SAP XI の `Input.Queue` にメッセージを

POST すると、pollForEvents() メソッドは次に適用できるメッセージを入力キューから検索します。メッセージは、進行中キューにステージされ、処理が完了するまでそこにとどまります。コネクターは、静的メタオブジェクトまたは動的メタオブジェクトを使用して、そのメッセージ・タイプ (つまり XML) がサポートされるかを検証します。次にコネクターは、構成済みデータ・ハンドラーにメッセージを渡し、データ・ハンドラーはそのメッセージを WebSphere Business Integration ビジネス・システムのビジネス・オブジェクトに変換します。設定される動詞は、このメッセージ・タイプに関して設定された変換プロパティーを反映しています。コネクターがビジネス・オブジェクトを InterChange Broker にデリバリーすると、そのメッセージは進行中キューから除去されます。

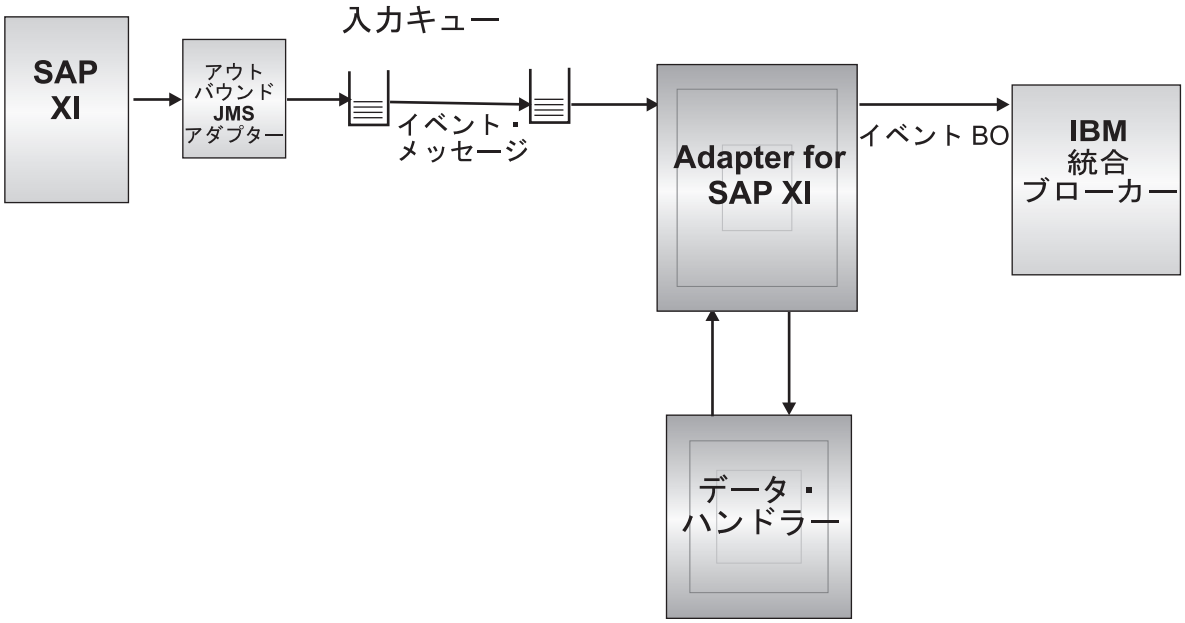

図 3. アプリケーションとコネクターの通信方式: イベント・デリバリー

保証付きイベント・デリバリー

保証付きイベント・デリバリー機能により、コネクター・フレームワークは、コネクターのイベント・ストアや JMS イベント・ストア、および宛先の JMS キューとの間で、イベントを失うことなく、また 2 回送信することなく、確実に送信することができます。JMS 対応にするには、コネクターの DeliveryTransport 標準プロパティーを JMS に構成する必要があります。このように構成すると、コネクターは JMS トランスポートを使用し、以降のコネクターと統合ブローカーの間のすべての通信は、このトランスポート経由で行われます。JMS トランスポートにより、メッセージは最終的には確実に宛先にデリバリーされます。その役割は、1 度トランザクション・キュー・セッションが開始したら、コミットが発行されるまでメッセージを確実にキャッシュに保持することです。障害が発生したり、ロールバックが発行された場合、メッセージは破棄されます。

注: 保証付きイベント・デリバリー機能を使用しない場合、コネクターがイベントをパブリッシュする時間 (コネクターが gotApplEvent() メソッドを自身の pollForEvents() メソッド内部で呼び出す時間) と、コネクターがイベント・レコ

ードを削除することによってイベント・ストアを更新する (または「イベント送付済み」状況を使用して更新する) 時間との間のわずかな期間に障害が発生する可能性があります。この期間に障害が発生すると、イベントは送信されますが、イベント・レコードは「進行中」状況でイベント・ストア内に残ります。コネクターは再始動時にイベント・ストア内に残ったイベント・レコードを検出して送信するため、結果的にイベントが 2 回送信されることになります。

JMS 対応のコネクターに対して保証付きイベント・デリバリー機能を構成するときに、JMS イベント・ストアを使用するように構成することも、不使用として構成することもできます。コネクターを保証付きイベント・デリバリー用に構成する方法については、「*コネクター開発ガイド (Java 用)*」の説明を参照してください。

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

イベント処理

イベント通知の場合、コネクターは、データベース・トリガーではなく、アプリケーションによってキューに書き込まれたイベントを検出します。イベントは、XI が XML メッセージを生成してそれを MQ メッセージ・キューに格納するときに発生します。

検索

コネクターは `pollForEvents()` メソッドを使用して、メッセージに対し、通常の間隔で MQ キューをポーリングします。メッセージを検出すると、コネクターは MQ キューからメッセージを検索し、それを検査してメッセージのフォーマットを判別します。フォーマットがコネクターの静的オブジェクトに定義されている場合、コネクターはメッセージ本文と、そのフォーマットに関連付けられたビジネス・オブジェクトの新規インスタンスの両方を、構成済みのデータ・ハンドラーに渡します。この場合データ・ハンドラーは、そのビジネス・オブジェクトを取り込んで、動詞を指定します。フォーマットが静的メタオブジェクトに定義されていない場合、コネクターは、メッセージ本文のみをデータ・ハンドラーに渡します。この場合データ・ハンドラーは、そのメッセージに対して適切なビジネス・オブジェクトを判別して作成し、取り込むことになります。イベントの失敗のシナリオについては、49 ページの『エラー処理』を参照してください。

コネクターがメッセージを処理するときには、最初に入力キューへのトランザクション・セッションが開始されます。トランザクションを使用するこの方法では、コネクターがビジネス・オブジェクトを正常にサブミットしたがキュー内でのトランザクションのコミットに失敗した場合に、ビジネス・オブジェクトが 1 つのコラボレーションに 2 回送信される可能性があります。この問題を回避するために、コネクターは、すべてのメッセージを進行中キューに移動します。進行中キューでは、メッセージは処理が完了するまで保持されます。処理中にコネクターの予期しないシャットダウンが発生した場合、進行中キュー内のメッセージは、元の入力キューに復元されずにそのまま保持されます。

注: JMS サービス・プロバイダーとのトランザクション・セッションでは、キュー内の要求されたアクションがすべて実行され、キューからイベントが除去される前にコミットされる必要があります。したがって、コネクターがキューからメッセージを検索するときは、1) メッセージがビジネス・オブジェクトに変換され、2) ビジネス・オブジェクトが `gotApplEvents()` メソッドによって InterChange Server にデリバリーされ、3) 戻り値が受信される、という 3 つの処理が完了してから、その検索をコミットします。

リカバリー

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

始動時の失敗

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

再処理

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

無視

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

Log error

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

アーカイブ

コネクターのプロパティー `ArchiveQueue` が指定され、それが有効なキューを示している場合、コネクターは、正常に処理されたすべてのメッセージのコピーをアーカイブ・キューに置きます。`ArchiveQueue` が定義されていない場合、メッセージは処理後に破棄されます。アンサブスクライブされたメッセージまたはエラーが含まれるメッセージのアーカイブについては、49 ページの『エラー処理』を参照してください。

注: JMS 規則により、検索されたメッセージを即時に他のキューに発行することはできません。メッセージをアーカイブして再デリバリーできるようにするた

め、コネクターはまず、オリジナルの本文およびヘッダー (該当する場合) を複写した第 2 のメッセージを生成します。JMS サービス・プロバイダーとの競合を避けるため、JMS に必須のフィールドのみが複写されます。したがって、フォーマット・フィールドは、アーカイブまたは再デリバリーされるメッセージ用にコピーされる唯一の追加メッセージ・プロパティーになります。

ビジネス・オブジェクト要求

ビジネス・オブジェクト要求は InterChange Server がビジネス・オブジェクトを `doVerbFor()` メソッドに送信したときに処理されます。コネクターは、構成済みデータ・ハンドラーを使用してビジネス・オブジェクトを WebSphere MQ メッセージに変換し、発行します。処理されるビジネス・オブジェクトのタイプに関する要件は、データ・ハンドラーにおける要件だけで、その他にはありません。

動詞の処理

コネクターは、コラボレーションから渡されたビジネス・オブジェクトを、各ビジネス・オブジェクトの動詞に基づいて処理します。コネクターはビジネス・オブジェクト・ハンドラーと `doForVerb()` メソッドを使用して、コネクターがサポートするビジネス・オブジェクトを処理します。コネクターは以下のビジネス・オブジェクトの動詞をサポートします。

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

注: Create、Update、および Delete 動詞を持つビジネス・オブジェクトは、同期でも非同期でも発行できます。デフォルト・モードは非同期です。コネクターは、Retrieve、Exists、または Retrieve by Content 動詞を持つビジネス・オブジェクトの非同期デリバリーはサポートしていません。したがって、Retrieve、Exists、または Retrieve by Content 動詞の場合、デフォルト (かつ唯一の) モードは同期です。

Create、Update および Delete

Create、Update、および Delete 動詞を持つビジネス・オブジェクトの処理は、そのオブジェクトの発行が同期か非同期かによって異なります。

非同期デリバリー

これは、Create、Update、および Delete 動詞を持つビジネス・オブジェクトのデフォルト・デリバリー・モードです。メッセージは、データ・ハンドラーを使用してビジネス・オブジェクトから作成され、出力キューに書き込まれます。メッセージがデリバリーされると、コネクターは `BON_SUCCESS` を戻し、それ以外の場合は `BON_FAIL` を戻します。

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

同期デリバリー

コネクターのプロパティーに replyToQueue が定義されていて、かつビジネス・オブジェクトの変換プロパティーに responseTimeout が存在する場合、コネクターは要求を同期モードで発行します。その後、コネクターは、受信側アプリケーションで適切なアクションが実行されたことを確認するため、応答を待ちます。

SAP Exchange Infrastructure に対して、コネクターは、次の表に示すヘッダーが付いたメッセージを最初に発行します。

フィールド	説明	値
Format	フォーマット名	変換プロパティーに定義された出力フォーマット。IBM 要件に合わせて 8 文字に切り捨てられます (例: MQSTR)。
MessageType	メッセージ・タイプ	MQMT_DATAGRAM* (受信側アプリケーションからの応答を期待しない場合) MQMT_REQUEST* (応答を期待する場合)
Report	要求されるレポート・メッセージのオプション	応答メッセージが期待される場合、このフィールドには以下の値が取り込まれます。MQRO_PAN*: 処理が成功したときに正のアクション・レポートを要求することを示す。MQRO_NAN*: 処理が失敗したときに負のアクション・レポートを要求することを示す。MQRO_COPY_MSG_ID_TO_CORREL_ID*: 生成されるレポートの相関 ID を、最初に発行した要求のメッセージ ID と同一とすることを示す。
ReplyToQueue	応答キューの名前	応答メッセージが期待される場合、このフィールドにはコネクターのプロパティー ReplyToQueue の値が取り込まれます。
Persistence	メッセージの永続性	MQPER_PERSISTENT*
Expiry	メッセージの存続時間	MQEI_UNLIMITED*

* は、IBM によって定義された定数を示す。

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

Report フィールドは、受信側アプリケーションから正と負の両方のアクション・レポートを期待することを示すように設定されます。メッセージを発行したスレッドは、受信側アプリケーションが要求を処理できたかどうかを示す応答メッセージを待ちます。

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

フィールド	説明	値
Format	フォーマット名	変換プロパティーに定義された busObj の入力フォーマット
MessageType	メッセージ・タイプ	MQMT_REPORT*

* は、IBM によって定義された定数を示す。

Verb	フィードバック・フィールド	メッセージ本文
Create、Update、または Delete	SUCCESS VALCHANGE	(オプション) 変更が反映された直列化ビジネス・オブジェクト。
	VALDUPES FAIL	(オプション) エラー・メッセージ。

WebSphere MQ フィードバック・コード	対応する CrossWorlds 応答*
MQFB_PAN または MQFB_APPL_FIRST	SUCCESS
MQFB_NAN または MQFB_APPL_FIRST + 1	FAIL
MQFB_APPL_FIRST + 2	VALCHANGE
MQFB_APPL_FIRST + 3	VALDUPES
MQFB_APPL_FIRST + 4	MULTIPLE_HITS
MQFB_APPL_FIRST + 5	FAIL_RETRIEVE_BY_CONTENT
MQFB_APPL_FIRST + 6	BO_DOES_NOT_EXIST
MQFB_APPL_FIRST + 7	UNABLE_TO_LOGIN
MQFB_APPL_FIRST + 8	APP_RESPONSE_TIMEOUT (コネクター・エージェントは即時終了します)

* 詳細は、「*コネクター開発ガイド (Java 用)*」を参照。

ビジネス・オブジェクトを処理できた場合、アプリケーションは、フィードバック・フィールドに MQFB_PAN (または特定の WebSphere Business Integration システム値) が設定されたレポート・メッセージを作成します。オプションでアプリケーションは、変更があればその変更を含めて直列化されたビジネス・オブジェクトをメッセージ本文に取り込みます。ビジネス・オブジェクトを処理できなかった場合、アプリケーションは、フィードバック・フィールドに MQFB_NAN (または特定の WebSphere Business Integration システム値) が設定されたレポート・メッセージを作成し、オプションで、エラー・メッセージをメッセージ本文に組み込みます。どちらの場合も、アプリケーションは、メッセージの correlationID フィールドにコネクター・メッセージの messageID を設定し、それを replyTo フィールドに指定されたキューに発行します。

応答メッセージを検索するとき、コネクターはデフォルトで、応答の correlationID を要求メッセージの messageID と突き合わせます。続いてコネクターは、要求を発行したスレッドに通知します。応答のフィードバック・フィールドに応じて、コネクターは、メッセージ本文にビジネス・オブジェクトまたはエラー・メッセージが含まれていると期待します。ビジネス・オブジェクトが期待されるにもかかわらずメッセージ本文に取り込まれていない場合、コネクターは、要求操作で InterChange Server から最初に発行されたものと同じビジネス・オブジェクトをそのまま戻します。エラー・メッセージが期待されるにもかかわらずメッセージ本文に取り込まれていない場合は、汎用エラー・メッセージが応答コードとともに InterChange Server に戻されます。ただし、メッセージ・セレクターを使用して、識別やフィルター操作を行ったり、アダプターが指定の要求に対する応答メッ

セージを識別する方法を制御することもできます。このメッセージ・セレクター機能は、JMS の機能です。この機能は、同期要求処理にのみ適用されます。詳細は以降で説明します。

メッセージ・セレクターを使用した応答メッセージのフィルター操作: 同期要求処理のビジネス・オブジェクトを受信すると、コネクターは、動詞のアプリケーション固有情報に response_selector ストリングがあるかどうかを検査します。response_selector が定義されていない場合、コネクターは、前述の相関 IDを使用して応答メッセージを識別します。

response_selector が定義されている場合、コネクターは、以下の構文で名前と値のペアが指定されていると期待します。

```
response_selector=JMSCorrelationID LIKE'selectorstring'
```

メッセージの selectorstring は応答を一意に識別でき、その値は次の例で示すように単一引用符で囲まれている必要があります。

```
response_selector=JMSCorrelationID LIKE 'Oshkosh'
```

この例では、要求メッセージを発行した後、アダプターは、correlationID が「Oshkosh」となっている応答メッセージが ReplyToQueue にあるかどうかをモニターします。アダプターは、このメッセージ・セレクターと一致する最初のメッセージを検索すると、それを応答としてディスパッチします。

オプションで、アダプターによるランタイム置換を実行して、要求ごとに固有のメッセージ・セレクターを生成することができます。その場合、メッセージ・セレクターの代わりに、整数を中括弧で囲んだ形式でプレースホルダーを指定します (例: '{1}')。次にコロンを付けて、置換に使用する属性をコンマで区切ってリストします。プレースホルダー内の整数は、置換に使用する属性の索引となります。例えば、メッセージ・セレクターを次のように指定します。

```
response_selector=JMSCorrelationID LIKE '{1}': MyDynamicMO.CorrelationID
```

この例は、アダプターに {1} をセレクターの後の最初の属性 (この場合は、子オブジェクト MyDynamicMO の属性 CorrelationId) の値に置き換えるよう通知します。属性 CorrelationID の値が 123ABC であれば、アダプターは、以下の基準で作成されたメッセージ・セレクターを生成し、使用します。

```
JMSCorrelation LIKE '123ABC'
```

これによって、応答メッセージを識別します。

次のように、複数の置換を指定することもできます。

```
response_selector=PrimaryId LIKE '{1}' AND AddressId LIKE '{2}' :
PrimaryId, Address[4].AddressId
```

この例では、アダプターは {1} をトップレベル・ビジネス・オブジェクトの属性 PrimaryId の値に置換し、{2} を子コンテナー・オブジェクト Address の 5 番目の位置にある AddressId の値に置換します。この方法を使用すれば、応答メッセージ・セレクター内にあるビジネス・オブジェクトおよびメタオブジェクトの任意の

属性を参照できます。Address[4].AddressId を使用した高度な検索の実行方法については、JCDK API マニュアル (getAttribute メソッド) を参照してください。

以下のいずれかの状況が発生した場合は、実行時にエラーが報告されます。

- '{}' シンボルの間に非整数を指定した場合
- 属性が定義されていない索引を指定した場合
- 指定された属性が、ビジネス・オブジェクトまたはメタオブジェクトに存在しない場合
- 属性パスの構文が誤っている場合

例えば、メッセージ・セレクターにリテラル値「{」または「}」を入れる場合は、それぞれ「{{」または「{}」を使用できます。また、これらの文字を属性値に置くこともできます。その場合、最初の「{」は不要です。エスケープ文字を使用した例を以下に示します。response_selector=JMSCorrelation LIKE '{1}' and CompanyName='A{{P': MyDynamicMO.CorrelationID

コネクターは、このメッセージ・セレクターを以下のように解決します。

```
JMSCorrelationID LIKE '123ABC' and CompanyName='A{P'
```

コネクターが属性値内で特殊文字 (「{」、「}」、「:」、「;」など) を検出すると、これらの文字は照会ストリングに直接挿入されます。これにより、アプリケーション固有情報の区切り文字でもある特殊文字を照会ストリングに組み込むことができます。

次の例は、リテラル・ストリングの置換が属性値からどのように抽出されるかを示します。

```
response_selector=JMSCorrelation LIKE '{1}' and CompanyName='A{{P':
MyDynamicMO.CorrelationID
```

MyDynamicMO.CorrelationID に値 {A:B}C;D が含まれる場合、コネクターはメッセージ・セレクターを以下のように解決します。 JMSCorrelationID LIKE '{A:B}C;D' and CompanyName='A{P'

応答セレクター・コードの詳細については、JMS 1.0.1 仕様を参照してください。

カスタム・フィードバック・コードの作成: WebSphere MQ フィードバック・コードがデフォルトの解釈をオーバーライドするように拡張するには、コネクターのプロパティー FeedbackCodeMappingMO を指定します。このプロパティーを使用すれば、すべての WebSphere Business Integration システム固有の戻り状況値が WebSphere MQ フィードバック・コードにマップされるメタオブジェクトを作成できます。(このメタオブジェクトを使用して) フィードバック・コードに割り当てられた戻り状況は、InterChange Server に渡されます。

Retrieve、Exists、および Retrieve By Content

Retrieve、Exists、および Retrieve By Content 動詞を持つビジネス・オブジェクトは、同期デリバリーのみをサポートします。コネクターは、これらの動詞を持つビジネス・オブジェクトを、Create、Update、および Delete 動詞に対して定義された同期デリバリーと同様に処理します。ただし、Retrieve、Exists、および Retrieve By

Content 動詞を使用する場合は、responseTimeout と replyToQueue が必要です。さらに、Retrieve By Content および Retrieve 動詞の場合、トランザクションを完了するためには、直列化されたビジネス・オブジェクトがメッセージ本文に取り込まれている必要があります。

次の表に、これらの動詞の応答メッセージを示します。

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

ロケール依存データの処理

コネクターは国際化され、2 バイト文字セットをサポートし、特定の言語でメッセージ・テキストを配信できるようになっています。コネクターがある文字コード・セットを使用する場所から別の文字コード・セットを使用する場所にデータを転送する場合、文字変換を行ってデータの意味を保持します。

Java 仮想マシン (JVM) 内での Java ランタイム環境は、Unicode 文字コード・セットでデータを表します。Unicode には、ほとんどの既知の文字コード・セット (1 バイト系と複数バイト系を含む) の文字に対応できるエンコード方式が組み込まれています。IBM WebSphere Business Integration システムのほとんどのコンポーネントは Java で書かれています。したがって、たいていの統合コンポーネント間のデータ転送の場合には、文字変換の必要はありません。

エラーおよび通知メッセージを国または地域に応じて適切な言語でログに記録するには、ご使用の環境に応じて Locale 標準構成プロパティーを構成します。構成プロパティーの詳細については、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。

共通の構成タスク

このセクションでは、ほとんどの開発者が実行する必要のあるいくつかの構成タスクと始動タスクについて概説します。

アダプターのインストール

アダプターのインストールの詳細については、第 2 章を参照してください。

コネクター・プロパティーの構成

コネクターの構成プロパティーには、標準構成プロパティーとコネクター固有の構成プロパティーという 2 つのタイプがあります。これらのプロパティーには、変更する必要のないデフォルト値を持つものもあります。コネクターを実行する前に、

これらのプロパティーの一部の値を設定する必要があります。詳しくは、『アダプターのインストールと構成』を参照してください。

SAP Exchange Infrastructure 用アダプターのコネクター・プロパティーを構成するときは、以下の点を確認してください。

- コネクター・プロパティー HostName に指定した値が、WebSphere MQ サーバーのホストの名前と一致している。
- コネクター・プロパティー Port に指定した値が、キュー・マネージャーのリスナーのポートと一致している。
- コネクター・プロパティー Channel に指定した値が、キュー・マネージャーのサーバー接続チャネルと一致している。
- コネクター・プロパティー InputQueue、InProgressQueue、ArchiveQueue、ErrorQueue、および UnsubscribeQueue のキュー URI が有効で、実際に存在する。

通知なしで要求を送信するようにコネクターを構成

通知なしで要求を送信するように (「発信後削除」とも呼ばれるデフォルトの非同期モード) コネクターを構成するには、次の手順を実行します。

- 送信する要求を表すビジネス・オブジェクトを作成します。このビジネス・オブジェクトは、コネクター用に構成したデータ・ハンドラーとの互換性がある必要があります。
- 静的メタオブジェクトまたは動的メタオブジェクトを使用して、ターゲット・キューとフォーマットを指定します。静的メタオブジェクトおよび動的メタオブジェクトの詳細については、第 2 章、『アダプターのインストールと構成』を参照してください。
- (静的または動的) メタオブジェクトのプロパティー ResponseTimeout を -1 に設定します。これによりコネクターは、戻りを確認せずにビジネス・オブジェクトを発行します。

要求を送信して通知を受け取るようにコネクターを構成

要求を送信して通知を受け取るように (同期イベント処理) コネクターを構成するには、以下の手順を実行します。

- 『通知なしで要求を送信するようにコネクターを構成』で説明した手順に従います。ただし ResponseTimeout には、コネクターが応答を待つ時間を示す正の値を指定してください。
- コネクターが応答メッセージで期待する内容の詳細については、8 ページの『Create、Update および Delete』を参照してください。リストされている要件と応答メッセージが合わない場合、コネクターはエラーを報告するか、あるいは応答メッセージの認識に失敗します。『メタオブジェクト属性の構成』のセクションも参照してください。

静的メタオブジェクトの構成

静的メタオブジェクトには、ビジネス・オブジェクトや、コネクターによるそれらの処理方法について指定したアプリケーション固有情報が含まれます。静的メタオ

ブジェクトはコネクターに対して、ビジネス・オブジェクトの処理に必要なすべての情報をコネクターの始動時に提供します。

実装時に、各種のビジネス・オブジェクトをどのキューに送信すべきかがわかっている場合は、静的メタオブジェクトを使用してください。このオブジェクトを作成し構成するには、以下の手順を実行します。

- 第 2 章の『静的メタオブジェクト』の手順に従います。
- コネクターが静的メタオブジェクトをサブスクライブするようにするため、コネクター固有プロパティー ConfigurationMetaObject に静的メタオブジェクトの名前を指定します。

動的メタオブジェクトの構成

コネクターによるビジネス・オブジェクトの処理がシナリオによって変わるようにする場合は、動的メタオブジェクトを使用してください。これは、ビジネス・オブジェクトに追加する子オブジェクトです。動的メタオブジェクトは、要求の処理方法を (実行時に) コネクターに知らせます。静的メタオブジェクトがビジネス・オブジェクトの処理に必要なすべての情報をコネクターに提供するのとは異なり、動的メタオブジェクトは、特定のシナリオの処理を行うために必要となる追加ロジックの部分のみを提供します。動的メタオブジェクトを作成し構成するには、以下の手順を実行します。

- 動的メタオブジェクトを作成し、それを子として要求ビジネス・オブジェクトに追加します。
- コネクターに発行する前に、動的メタオブジェクトにターゲット・キュー、メッセージ・フォーマットなどの情報を取り込む追加ロジックを、コラボレーションに組み込みます。

コネクターは動的メタオブジェクトを検査し、その情報を使用してビジネス・オブジェクトの処理方法を決定します。詳しくは、『動的メタオブジェクト』を参照してください。

MQMD フォーマットの構成

MQMD はメッセージ記述子です。MQMD は、アプリケーションから別のアプリケーションにメッセージが移動するときに、アプリケーション・データを伴った制御情報を格納します。静的メタオブジェクトまたは動的メタオブジェクトの MQMD 属性 OutputFormat に値を指定する必要があります。

キュー URI の構成

SAP Exchange Infrastructure 用アダプターで使用できるようにキューを構成するには、以下の手順を実行します。

- すべてのキューを URI (Uniform Resource Identifier) で指定します。構文は以下のとおりです。

  queue://<*queue manager name*>/<*actual queue*>

- コネクター固有の構成プロパティーにキュー・マネージャーのホストを指定します。
- ターゲット・アプリケーションが MQMD ヘッダーのみを期待しており、JMS クライアントで使用される拡張 MQRFH2 ヘッダーを処理できない場合は、キュー

URI の後ろに ?targetClient=1 を付けます。詳しくは、「WebSphere MQ Application Programming Guide」を参照してください。

データ・ハンドラーの構成

データ・ハンドラーの構成方法は、次の 2 つがあります。

- コネクター固有のプロパティー DataHandlerClassName でデータ・ハンドラーのクラス名を指定します。
- あるいは、MIME タイプと、その MIME タイプの構成を定義するデータ・ハンドラー・メタオブジェクトを、コネクター固有のプロパティー DataHandlerMimeType および DataHandlerConfigMO にそれぞれ指定します。詳しくは、「*データ・ハンドラー・ガイド*」を参照してください。

始動スクリプトの変更

コネクターの始動方法については、第 2 章を参照してください。始動前にコネクターのプロパティーを構成する必要があります。また、始動ファイルを変更する必要があります。

- クライアント・ライブラリーのロケーションを指すように、start_connector スクリプトを必ず変更してください。複数のバージョンのクライアント・ライブラリーや、ご使用の WebSphere MQ サーバーで最新ではないバージョンをインストールすることはしないでください。

第 2 章 アダプターのインストールと構成

この章では、アダプターをインストールおよび構成する方法と、アダプターと連動するようにメッセージ・フローを構成する方法について説明します。

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。SAP Exchange Infrastructure のアダプターのバージョン 1.0.0 は、以下のアダプター・フレームワークと統合ブローカーでサポートされています。

- アダプター・フレームワーク: WebSphere Business Integration Adapter Framework、バージョン 2.4.x.
- 統合ブローカー:
  - WebSphere InterChange Server、バージョン 4.1.1 および 4.2.x
  - WebSphere Application Server Enterprise、バージョン 5.0.1 (WebSphere Studio Application Developer Integration Edition、バージョン 5.0 と併用)

例外については、「*リリース情報*」を参照してください。

統合ブローカーおよびその前提条件のインストールに関する説明については、以下の文書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。
- メッセージ・ブローカー (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) については、「*WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド*」およびご使用のメッセージ・ブローカーのインストール資料を参照してください。これらの資料の一部は、Web サイト http://www.ibm.com/software/integration/mqfamily/library/manualsa にあります。

- WebSphere Application Server については、「*アダプター実装ガイド (WebSphere Application Server)*」および http://www.ibm.com/software/webservers/apserv/library.html の資料を参照してください。

必要なソフトウェア

Adapter for SAP XI は、以下のプラットフォームで稼動します。

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

コネクターは、WebSphere MQ 5.1、5.2 を通じてアプリケーションとのインターオペラビリティーをサポートします。

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

さらに、IBM WebSphere MQ Java クライアント・ライブラリーが必要です。

SAP XI アダプターのインストール

WebSphere Business Integration アダプター製品のインストールについては、次のサイトで WebSphere Business Integration Adapters Infocenter にある「*WebSphere Business Integration Adapters インストール・ガイド*」を参照してください。

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

インストールされたファイル構造のナビゲート

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

注: WebSphere MQ および JMS は、Windows 環境でも UNIX 環境でも、通常別のディレクトリーにインストールされます。例えば AIX システムでは、デフォルトでは WebSphere MQ は `/var/mqm/` にインストールされますが、JMS は `/usr/mqm/java/lib` にインストールされます。JMS のインストールを `/var/mqm/java/lib` に指定変更して、日常的な `/usr` 関連のシステム管理タスクによって削除されないようにすることも可能です。Windows 上でも同様であり、WebSphere MQ は一般に `¥Program Files¥WebSphere MQ` にインストールされ、JMS は `¥Program Files¥IBM¥MQSerires¥Java` にインストールされます。それに応じて、WebSphere MQ コネクター始動スクリプト内のクラスパスを更新します。

Windows のファイル構造

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

ユーティリティーによって、コネクターが *ProductDir*¥connectors¥WebSphereMQConnector ディレクトリーにインストールされ、コネクターのショートカットが「スタート」メニューに追加されます。

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

ProductDir のサブディレクトリー	説明
connectors¥WebSphereMQ¥CWWebSphereMQ.jar	WebSphere MQ コネクターのみが使用するクラスを含みます。
connectors¥WebSphereMQConnector¥start_WebSphereMQ.bat	コネクターの始動スクリプト (NT/2000)。
connectors¥messages¥WebSphereMQConnector.txt	コネクターのメッセージ・ファイル。
repository¥WebSphereMQ¥CN_WebSphereMQ.txt	コネクターのリポジトリー定義。
connectors¥WebSphereMQ¥samples¥LegacyContact¥WebSphereMQConnector.cfg	WebSphere MQ 構成ファイルのサンプル
connectors¥WebSphereMQ¥samples¥LegacyContact¥PortConnector.cfg	Port コネクター構成ファイルのサンプル
connectors¥WebSphereMQ¥samples¥LegacyContact¥Sample_WebSphereMQ _LegacyContact.xsd	スキーマのサンプル
connectors¥WebSphereMQ¥samples¥LegacyContact¥Sample_WebSphereMQ _MO_Config.xsd	メタオブジェクトのサンプル
connectors¥WebSphereMQ¥samples¥LegacyContact¥Sample_WebSphereMQ _MO_DataHandler.xsd	データ・ハンドラー・メタオブジェクトのサンプル
connectors¥WebSphereMQ¥samples¥LegacyContact¥Sample_WebSphereMQ _MO_DataHandler_DelimitedConfig.xsd	区切られたデータ・ハンドラー・メタオブジェクトのサンプル
connectors¥WebSphereMQ¥samples¥LegacyContact¥Sample_WebSphereMQ _DynMO_Config.xsd	動的なメタオブジェクトのサンプル
connectors¥WebSphereMQ¥samples¥LegacyContact¥JMSPropertyPairs.xsd	JMS プロパティーのサンプル

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

UNIX のファイル構造

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

このユーティリティーにより、コネクターが *ProductDir*/connectors/WebSphereMQConnector ディレクトリーにインストールされます。

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

ProductDir のサブディレクトリー	説明
connectors/WebSphereMQ/CWWebSphere MQ.jar	SAP Exchange Infrastructure コネクターのみが使用するクラスを含みます。
connectors/WebSphereMQ/start_WebSphereMQ.sh	コネクターのシステム始動スクリプト。このスクリプトは、汎用のコネクター・マネージャー・スクリプトから呼び出されます。System Manager の「Connector Configuration」画面をクリックすると、インストーラーは、このコネクター・マネージャー・スクリプト用にカスタマイズされたラッパーを作成します。このカスタマイズされたラッパーを使用してコネクターを始動および停止してください。
connectors/messages/WebSphereMQ/Connector.txt	コネクターのメッセージ・ファイル。
repository/WebSphereMQ/CN_WebSphere MQ.txt	コネクターのリポジトリー定義。
connectors/WebSphereMQ/samples/LegacyContact/WebSphereMQConnector.cfg	WebSphere MQ 構成ファイルのサンプル
connectors/WebSphereMQ/samples/LegacyContact/PortConnector.cfg	Port コネクター構成ファイルのサンプル

ProductDir のサブディレクトリー	説明
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _LegacyContact.xsd	スキーマのサンプル
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _MO_Config.xsd	メタオブジェクトのサンプル
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _MO_DataHandler.xsd	データ・ハンドラー・メタオブジェクトのサンプル
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _MO_DataHandler_DelimitedConfig.xsd	区切られたデータ・ハンドラー・メタオブジェクトのサンプル
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _DynMO_Config.xsd	動的なメタオブジェクトのサンプル
connectors/WebSphereMQ/samples/LegacyContact/JMSPropertyPairs.xsd	JMS プロパティーのサンプル

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

コネクターの構成

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

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

- Connector Configurator の詳細および操作手順については、75 ページの『付録 A. Connector Configurator』を参照してください。
- 標準のコネクター・プロパティーについては、75 ページの『付録 A. Connector Configurator』を参照してください。
- コネクター固有のプロパティーについては、21 ページの『コネクター固有のプロパティー』を参照してください。

コネクターは、始動時に構成値を取得します。実行時セッション中に、1 つ以上のコネクター・プロパティーの値を変更することができます。AgentTraceLevel のように、変更がすぐに有効になるコネクター構成プロパティーもあります。変更後にコンポーネントの再始動やシステム再始動が必要なコネクター・プロパティーもあります。プロパティーが動的 (すぐに有効になる) か静的 (コネクター・コンポーネントの再始動やシステム再始動が必要) かを判別する方法については、Connector Configurator の「コネクター・プロパティー (Connector Properties)」ウィンドウの「更新メソッド」列を参照してください。

標準コネクター・プロパティー

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

注: Connector Configurator 内の構成プロパティーを設定するときに、BrokerType プロパティーによって使用するブローカーを指定します。このプロパティーが設定されると、ご使用のブローカーに関連するプロパティーが「Connector Configurator」ウィンドウに表示されます。

コネクター固有のプロパティー

コネクター固有の構成プロパティーは、コネクターが実行時に必要とする情報を提供します。また、コネクター固有のプロパティーを使用すれば、エージェントを再コーディングまたは再ビルドせずに、コネクター内の静的情報やロジックを変更できます。

以下の表に、コネクターのアダプター固有構成プロパティーを示します。プロパティーの説明については、以下の各セクションを参照してください。

Name	指定可能な値	デフォルト値	必要
ApplicationPassword	*ログイン・パスワード*		なし
ApplicationUserName	*ログイン・ユーザー ID*		なし
ArchiveQueue	*正常に処理されたメッセージのコピーが送信されるキュー*	queue://crossworlds.queuemanager/MQCONN.ARCHIVE	なし
CCSID	*キュー・マネージャー接続用文字セット*	ヌル	なし
Channel	*MQ サーバー・コネクター・チャネル*		可
ConfigurationMetaObject	*構成メタオブジェクトの名前*		可
DataHandlerClassName	データ・ハンドラー・クラス名	com.crossworlds.DataHandlers.text.xml	なし
DataHandlerConfigMO	データ・ハンドラー・メタオブジェクト	MO_DataHandler_ Default	可
DataHandlerMimeType	ファイルの *MIME タイプ*	text/xml	なし
DefaultVerb	*コネクターがサポートする動詞*	Create	
ErrorQueue	*未処理のメッセージのキュー*	queue://crossworlds.queuemanager/MQCONN.ERROR	なし
FeedbackCodeMappingMO	フィードバック・コード・メタオブジェクト		なし
HostName	*WebSphere MQ サーバー*		可
InDoubtEvents	FailOnStartup Reprocess IgnoreLogError	Reprocess	なし
InputQueue	*ポーリング・キュー*	queue://crossworlds.queuemanager/MQCONN.IN	なし
InProgressQueue	*進行中イベント・キュー*	queue://crossworlds.queuemanager/MQCONN.IN_PROGRESS	なし
PollQuantity	*InputQueue プロパティーで指定された各キューから検索するメッセージの数*	1	なし
Port	*WebSphere MQ リスナーのために確立するポート*		可
ReplyToQueue	*コネクターが要求を発行したときに応答メッセージが引き渡されるキュー*	queue://crossworlds.queuemanager/MQCONN.REPLYTO	なし
UnsubscribedQueue	*アンサブスクライブされたメッセージが送信されるキュー*	queue://crossworlds.queuemanager/MQCONN.UNSUBSCRIBE	なし
UseDefaults	true または false	false	

ApplicationPassword

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

デフォルト = なし。

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

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

デフォルト = なし。

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

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

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

CCSID
キュー・マネージャー接続の文字セット。プロパティーの値は、キュー URI 内の CCSID プロパティーの値と一致している必要があります。 30 ページの『キューのユニフォーム・リソース ID の理解』を参照してください。

デフォルト = ヌル。

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

デフォルト = なし。

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

ConfigurationMetaObject
コネクターの構成情報を含む静的なメタオブジェクトの名前。

デフォルト = なし。

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

デフォルト = com.crossworlds.DataHandlers.text.xml

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

デフォルト = MO_DataHandler_Default

DataHandlerMimeType
特定の MIME タイプに基づくデータ・ハンドラーを要求することができます。

デフォルト = text/xml

DefaultVerb
着信ビジネス・オブジェクト内部にその動詞が設定されるように指定します (ポーリング時にデータ・ハンドラーによって設定されなかった場合)。

デフォルト = Create

ErrorQueue
処理されなかったメッセージが送信されるキューです。

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

FeedbackCodeMappingMO
メッセージの受信を同期して InterChange Server へ応答する場合に使用するデフォルトのフィードバック・コードをオーバーライドし、再割り当てすることができます。このプロパティーでは、各属性名を認識してフィードバック・コードを表すメタオブジェクトを指定できます。フィードバック・コードの対応する値は、InterChange Server に渡される戻り状況です。コネクターは、WebSphere MQ 固有のフィードバック・コードを表す以下の属性値を受け入れます。

- MQFB_APPL_FIRST
- MQFB_APPL_FIRST_OFFSET_*N (ここで N は整数であり、*MQFB_APPL_FIRST + *N の値として解釈されます。)*
- MQFB_NONE
- MQFB_PAN
- MQFB_NAN

コネクターがメタオブジェクト内の属性値として受け入れる WebSphere business integration システム固有の状況コードは、以下のとおりです。

- SUCCESS
- FAIL
- APP_RESPONSE_TIMEOUT
- MULTIPLE_HITS
- UNABLE_TO_LOGIN
- VALCHANGE
- VALDUPES

以下の表に、メタオブジェクトのサンプルを示します。

属性名	デフォルト値
MQFB_APPL_FIRST	SUCCESS
MQFB_APPL_FIRST + 1	FAIL
MQFB_APPL_FIRST + 2	UNABLE_TO_LOGIN

デフォルト = なし。

HostName
WebSphere MQ のホストであるサーバーの名前です。

デフォルト = なし。

InDoubtEvents
コネクターの予期しないシャットダウンのために、処理が完了していない進行中イベントの処理方法を指定します。初期化中に進行中キューにイベントが見つかった場合に実行するアクションを、以下の 4 つから選択してください。

- FailOnStartup: エラー・ログを記録して即時にシャットダウンします。
- Reprocess: 残りのイベントを先に処理してから、入力キューのメッセージを処理します。
- Ignore: 進行中キューのすべてのメッセージを無視します。
- LogError: エラー・ログを記録しますが、シャットダウンはしません。

デフォルト = Reprocess。

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

InputQueue プロパティーが提供されていない場合も、コネクターは正常に始動しますが、警告メッセージが出力され、要求処理のみが実行されます。イベント処理は実行されません。

コネクターはラウンドロビン方式でキューをポーリングし、各キューから pollQuantity の数を上限としてメッセージを取得します。例えば、*pollQuantity* が 2 であるときに、MyQueueA に 2 つのメッセージ、MyQueueB に 1 つのメッセージ、および MyQueueC に 5 つのメッセージが含まれる場合、コネクターは以下のような方法でメッセージを取得します。

PollQuantity が 2 であるため、コネクターは pollForEvents を 1 回呼び出すごとに各キューから最大 2 つのメッセージを取得します。第 1 サイクル (2 回のうちの 1 回目) では、コネクターは MyQueueA、MyQueueB、および MyQueueC のそれぞれから先頭のメッセージを取得します。これでポーリングの第 1 ラウンドは完了します。PollQuantity が 1 であれば、コネクターはここで停止します。PollQuantity が 2 となっているため、コネクターはポーリングの第 2 ラウンド (2 回のうちの 2 回目) を開始し、MyQueueA と MyQueueC からそれぞれ 1 つずつメッセージを取得します。このとき MqQueueB は空であるため、スキップします。すべてのキューをそれぞれ 2 回ずつポーリングすると、メソッド pollForEvents への呼び出しが完了します。メッセージ検索の順序は以下のようになります。

1. MyQueueA から 1 つのメッセージ
2. MyQueueB から 1 つのメッセージ
3. MyQueueC から 1 つのメッセージ
4. MyQueueA から 1 つのメッセージ

5. MyQueueB は空なのでスキップ
6. MyQueueC から 1 つのメッセージ

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

InProgressQueue
処理中にメッセージを保持するメッセージ・キュー。System Manager を使用してコネクター固有のプロパティーからデフォルトの InProgressQueue 名を削除することによって、このキューなしで動作するようコネクターを構成できます。そのように設定すると、イベントの保留中にコネクターがシャットダウンされたときにイベント・デリバリーが影響を受ける場合があるという警告のプロンプトが、始動時に出されます。

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

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

デフォルト = 1

Port
WebSphere MQ リスナーのために確立するポート。

デフォルト = なし。

ReplyToQueue
コネクターが要求を発行したときに応答メッセージが送信されるキュー。動的な子メタオブジェクト内の属性を使用して、応答を無視することができます。これらの属性の詳細については、42 ページの『JMS ヘッダー、WebSphere MQ メッセージ・プロパティー、および動的な子メタオブジェクトの属性』を参照してください。

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

UnsubscribedQueue
サブスクライブされていないメッセージが送信されるキュー。

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

注: *誤っている場合や不明な場合があるため、WebSphere MQ が提供する値を常にチェックしてください。問題がある場合は、暗黙的に値を指定してください。

UseDefaults
Create 操作では、UseDefaults が true に設定されている場合、コネクターは、各ビジネス・オブジェクト属性に有効な値またはデフォルト値が指定されているかをチェックします。値が指定されている場合には、Create は正常に実行されます。パラメーターが false に設定されていると、コネクターは有効な値のみチェックし、値が設定されていなければ、Create 処理は失敗します。デフォルト値は false です。

保証付きイベント・デリバリーの使用可能化

以下のいずれかの方法によって、JMS 対応のコネクターに合わせて保証付きイベント・デリバリー機能を構成することができます。

- コネクターが (JMS ソース・キューとして実装された) JMS イベント・ストアを使用する場合、コネクター・フレームワークは JMS イベント・ストアを管理することができます。
- コネクターが JMS 以外のイベント・ストア (例えば JDBC 表、E メールのメールボックス、またはフラット・ファイルなど) を使用する場合、コネクター・フレームワークは JMS モニター・キューを使用して重複イベントが発生しないようにすることができます。

JMS イベント・ストアを使用するコネクターの保証付きイベント・デリバリー

JMS 対応のコネクターが JMS キューを使用してイベント・ストアを実装する場合、コネクター・フレームワークは「コンテナー」として機能し、JMS イベント・ストア (JMS ソース・キュー) を管理することができます。単一の JMS トランザクションで、コネクターは、ソース・キューからメッセージを削除し、宛先キューに入れることができます。このセクションでは、JMS イベント・ストアを含む JMS 対応のコネクターでの保証付きイベント・デリバリー機能の使用について説明します。

JMS イベント・ストアを使用するコネクター用の機能の使用可能化: JMS イベント・ストアを含む JMS 対応のコネクター用に保証付きイベント・デリバリー機能を使用可能にするには、コネクター構成プロパティーを表 1 に示す値に設定します。

表 1. JMS のイベント・ストアを使用するコネクター用の保証付きイベント・デリバリー機能関連のコネクター・プロパティー

コネクター・プロパティー	値
DeliveryTransport	JMS
ContainerManagedEvents	JMS
PollQuantity	イベント・ストアの 1 回のポーリングで処理するイベントの数。

表 1. JMS のイベント・ストアを使用するコネクター用の保証付きイベント・デリバリー機能関連のコネクター・プロパティー (続き)

コネクター・プロパティー	値
SourceQueue	コネクター・フレームワークがポーリングし、処理対象のイベントを取得する JMS ソース・キュー (イベント・ストア) の名前。**注:** ソース・キューおよびその他の JMS キューは、同じキュー・マネージャーに属していなければなりません。コネクターのアプリケーションが、別のキュー・マネージャーに格納されるイベントを生成する場合、リモート・キュー・マネージャーでリモート・キュー定義を指定する必要があります。その後、WebSphere MQ は、リモート・キューから、JMS 対応のコネクターが統合ブローカーへの転送に使用するキュー・マネージャーに、イベントを転送することができるようになります。リモート・キュー定義の構成方法については、IBM WebSphere MQ の資料を参照してください。

コネクターを構成するだけでなく、JMS ストア内のイベントおよびビジネス・オブジェクト間で変換を行うデータ・ハンドラーも構成する必要があります。このデータ・ハンドラー情報は、表 2 にまとめられているコネクター構成プロパティーで構成されています。

表 2. 保証付きイベント・デリバリーのデータ・ハンドラー・プロパティー

データ・ハンドラー・プロパティー	値	必須ですか?
MimeType	データ・ハンドラーが処理する MIME タイプこの MIME タイプは、呼び出すデータ・ハンドラーを示します。	可
DHClass	データ・ハンドラーを実装する Java クラスの絶対パス名	可
DataHandlerConfigMOName	MIME タイプとそのデータ・ハンドラーを関連付けるトップレベル・メタオブジェクトの名前	オプショナル

注: データ・ハンドラー構成プロパティーは、コネクター構成ファイル内に、他のコネクター構成プロパティーと共に存在します。

JMS イベント・ストアを使用するコネクターを、保証付きイベント・デリバリーを使用するように構成する場合、表 1 および表 2 の説明に従ってコネクター・プロパティーを設定する必要があります。これらのコネクター構成プロパティーを設定するには、Connector Configurator ツールを使用します。Connector Configurator は、「標準のプロパティー」タブで、表 1 のコネクター・プロパティーを表示します。そして、「データ・ハンドラー」タブで、表 2 のコネクター・プロパティーを表示します。

注: Connector Configurator は、DeliveryTransport コネクター構成プロパティーが JMS に設定され、ContainerManagedEvents が JMS に設定されている場合のみ、「データ・ハンドラー」タブのフィールドをアクティブにします。

イベント・ポーリングへの影響: ContainedManagedEvents を JMS に設定して、コネクターで保証付きイベント・デリバリー機能を使用すると、この機能を使用しない場合と比べて、コネクターの振る舞いが多少変化します。コンテナー管理イベントを提供するため、コネクター・フレームワークは以下のステップを実行してイベント・ストアをポーリングします。

1. JMS トランザクションを開始します。
2. イベント・ストアから JMS メッセージを読み取ります。

   イベント・ストアは JMS ソース・キューとして実装されます。JMS メッセージには、イベント・レコードが含まれます。JMS ソース・キューの名前は、SourceQueue コネクター構成プロパティーから取得します。
3. データ・ハンドラーを呼び出して、イベントをビジネス・オブジェクトに変換します。

   コネクター・フレームワークは、表 2 のプロパティーを用いて構成されたデータ・ハンドラーを呼び出します。
4. 統合ブローカーが WebSphere MQ Integrator Broker の場合、ビジネス・オブジェクトを、構成されたワイヤー・フォーマット (XML) に基づくメッセージに変換します。
5. 変換されたメッセージを JMS 宛先キューに送信します。
   WebSphere ICS 統合ブローカーを使用している場合、JMS 宛先キューに送信されるメッセージはビジネス・オブジェクトです。WebSphere MQ Integrator Broker を使用している場合、JMS 宛先キューに送信されるメッセージは XML メッセージ (データ・ハンドラーが生成したもの) です。
6. JMS トランザクションをコミットします。

   JMS トランザクションをコミットすると、メッセージは、同じトランザクションで JMS 宛先キューに書き込まれ、JMS ソース・キューから削除されます。
7. ステップ 1 から 6 を繰り返し実行します。PollQuantity コネクター・プロパティーは、このループを繰り返す回数を決定します。

重要: ContainerManagedEvents プロパティーを JMS に設定しているコネクターは、イベント・ポーリング実行のために pollForEvents() メソッドを*呼び出しません*。コネクターの基本クラスに pollForEvents() メソッドが含まれる場合、このメソッドは*呼び出されません*。

JMS 以外のイベント・ストアを使用するコネクターの保証付きイベント・デリバリー: JMS 対応のコネクターが、JMS 以外のソリューションを使用してイベント・ストア (JDBC イベント表、E メールのメールボックス、またはフラット・ファイルなど) を実装する場合、コネクター・フレームワークは、重複イベント除去を使用して重複イベントが発生しないようにすることができます。このセクションでは、JMS 以外のイベント・ストアを使用する JMS 対応のコネクターでの保証付きイベント・デリバリー機能の使用について説明します。

JMS 以外のイベント・ストアを使用するコネクターの機能の使用可能化: JMS 以外のイベント・ストアを使用する JMS 対応のコネクターで保証付きイベント・デ

リバリー機能を使用可能にするには、コネクター構成プロパティーを表 3 に示す値に設定する必要があります。

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

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

保証付きイベント・デリバリーを使用するようにコネクターを構成する場合、表 3 での説明に従ってコネクター・プロパティーを設定する必要があります。これらのコネクター構成プロパティーを設定するには、Connector Configurator ツールを使用します。このツールを使用すると、これらのコネクター・プロパティーが「標準のプロパティー」タブに表示されます。Connector Configurator の詳細については、付録 A を参照してください。

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

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

- これら 2 つの ObjectEventId が同じであれば、現在のビジネス・オブジェクトが重複イベントであるということになります。このような場合、コネクター・フレームワークは、現在のビジネス・オブジェクトが表すイベントを無視します。つまり、このイベントを統合ブローカーに*送信しません*。
- これらの ObjectEventId が*同じでない* 場合、ビジネス・オブジェクトは重複イベントではありません。この場合、コネクター・フレームワークは、現行のビジネス・オブジェクトを JMS モニター・キューにコピーしてから、それを JMS デリバリー・キューに送信します。これらはすべて同じ JMS トランザクションの一部として実行されます。JMS デリバリー・キューの名前は、DeliveryQueue コネクター構成プロパティーから取得します。gotApplEvent() メソッドを呼び出した後は、制御はコネクターの pollForEvents() メソッドに戻ります。

JMS 対応のコネクターが重複イベント除去をサポートするためには、コネクターの pollForEvents() メソッドに以下のステップが含まれるようにする必要があります。

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

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

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

キューのユニフォーム・リソース ID の理解

キューのユニフォーム・リソース ID (URI) は、先頭が queue:// で、その後に以下の要素が続きます。

- キューが存在するキュー・マネージャーの名前
- もう 1 つの /
- キューの名前
- 必要に応じて、残りのキュー・プロパティーを設定する名前と値のペアのリスト

例えば次の URI を使用すると、キュー・マネージャー crossworlds.queue.manager 上のキュー IN に接続し、すべてのメッセージが優先度 5 の WebSphere MQ メッセージとして送信されます。

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

以下の表に、キューの URI のプロパティー名を示します。

プロパティー名	説明	値
expiry	メッセージの存続時間 (ミリ秒)。	0 = 無制限。正整数 = タイムアウト (ミリ秒単位) 。
priority	メッセージの優先順位。	0 から 9 で、1 が最高優先順位を表します。値 -1 は、このプロパティーがキューの構成によって決定されることを意味します。値 -2 は、コネクターが独自のデフォルト値を使用可能であることを指定します。

プロパティー名	説明	値
persistence	メッセージをディスクに「永久保存」すべきかどうかを指定。	1 = 非永続、2 = 永続。値 -1 は、このプロパティーがキューの構成によって決定されることを意味します。値 -2 は、コネクターが独自のデフォルト値を使用可能であることを指定します。
CCSID	アウトバウンド・メッセージの文字セット・エンコード方式。	整数 (基本の WebSphere MQ 資料でリストされている有効な値)。この値は、CCSID コネクター固有構成プロパティーの値と一致していなければなりません。
targetClient	受信側アプリケーションが JMS に準拠しているかどうかを指定。	0 = JMS (MQRFH2 ヘッダー)、1 = MQ (MQMD ヘッダーのみ)。
encoding	数値フィールドを表す方法。	基本 WebSphere MQ 資料に記載されている整数値。

注: アダプターは、MQMessages 内のデータの文字セット (CCSID) またはエンコード属性を制御することはできません。データをメッセージ・バッファーとの間でやり取りするときにデータ変換が適用されるため、コネクターは IBM WebSphere MQ 用に実装された JMS に依存してデータを変換します (IBM WebSphere MQ Java クライアント・ライブラリーの資料を参照してください)。したがって、これらの変換は、ネイティブ WebSphere MQ API が オプション MQGMO_CONVERT を使用して実行する変換と、双方向で等価でなければなりません。コネクターは、変換処理の差異や失敗を制御することはできません。 コネクターは、追加的な変更を加えることなく、WebSphere MQ がサポートする CCSID またはエンコードのメッセージ・データを検索できます。特定の CCSID またはエンコード方式のメッセージを送信するには、出力キューは完全修飾 URI でなければならず、CCSID および encoding の値を指定する必要があります。コネクターはこの情報を WebSphere MQ に渡します。WebSphere MQ は、MQMessage のデリバリーのためにデータをエンコードするときに (JMS APIを通じて) この情報を使用します。CCSID およびエンコードがサポートされていないという問題は、多くの場合、IBM の Web サイトから最新バージョンの IBM WebSphere MQ Java クライアント・ライブラリーをダウンロードすることによって解決できます。CCSID およびエンコード方式に固有の問題が解決できない場合は、WebSphere business integration システムのテクニカル・サポートに連絡して、代わりの Java 仮想マシンを使用してコネクターを実行できるどうかお問い合わせください。

メタオブジェクト属性の構成

SAP Exchange Infrastructure 用のコネクターは、以下の 2 種類のメタオブジェクトを認識し、読み取ることができます。

- 静的なコネクター・メタオブジェクト
- 動的な子メタオブジェクト

動的な子メタオブジェクトの属性値は、静的なメタオブジェクトの属性値を複製して、オーバーライドします。

静的メタオブジェクト

SAP Exchange Infrastructure の静的なメタオブジェクトは、各種のビジネス・オブジェクト用に定義された変換プロパティーのリストで構成されています。ビジネス・オブジェクトの変換プロパティーを定義するには、まずストリング属性を作成し、構文 `busObj_verb` を用いて名前を付けます。例えば、動詞 Create 付きの Customer オブジェクトの変換プロパティーを定義するには、`Customer_Create` という名前の属性を作成します。属性のアプリケーション固有のテキストの中で、実際の変換プロパティーを指定します。

注: 静的なメタオブジェクトが指定されていない場合、コネクターは、ポーリング時に所定のメッセージ・フォーマットを特定のビジネス・オブジェクト・タイプにマップできません。このような場合、コネクターは、ビジネス・オブジェクトを指定せずにメッセージ・テキストを構成済みのデータ・ハンドラーに渡します。データ・ハンドラーがテキストのみに基づいてビジネス・オブジェクトを作成できない場合、コネクターは、このメッセージ・フォーマットが認識できないことを示すエラーを報告します。

以下の表では、メタオブジェクト・プロパティーについて説明します。

プロパティー名	説明
`CorrelationID`	このプロパティーは、要求処理の間のみアダプターの振る舞いに影響し、動的メタオブジェクト内の `CorrelationID` と同様に処理されます。詳細については、46 ページの『Asynchronous 要求処理』を参照してください。
`CollaborationName`	`CollaborationName` は、ビジネス・オブジェクトと動詞の組み合わせ用の属性の、アプリケーション固有のテキストで指定する必要があります。例えば、Create 動詞付きのビジネス・オブジェクト Customer の同期要求を処理しようとする場合、静的メタデータ・オブジェクトには `Customer_Create` という名前の属性が含まれていなければなりません。 `Customer_Create` 属性には、名前と値のペアを含むアプリケーション固有のテキストが含まれている必要があります。例: CollaborationName=MyCustomerProcessingCollab。構文について詳しくは、35 ページの『アプリケーション固有の情報』を参照してください。 このようにしなかった場合、コネクターが同期的に Customer ビジネス・オブジェクトにかかわる要求の処理を試みたときに、ランタイム・エラーが発生します。 **注:** このプロパティーは、同期要求に対してのみ使用可能です。

プロパティー名	説明
DataEncoding	DataEncoding は、メッセージの読み取り/書き込みに使用されるエンコード方式です。静的なメタオブジェクト内にこのプロパティーが指定されていない場合、コネクターは特定のエンコード方式を使用せずにメッセージの読み取りを試みます。DataEncoding が動的な子メタオブジェクトで定義されている場合、静的なメタオブジェクトで定義されている値はオーバーライドされます。デフォルト値は Text です。この属性の値のフォーマットは、messageType[:*enc*] です。すなわち、Text:ISO8859_1、Text:UnicodeLittle、Text、または Binary です。このプロパティーは、内部的に InputFormat プロパティーに関連付けられ、InputFormat ごとに DataEncoding を 1 つのみ指定します。
DataHandlerConfigMO	構成情報を提供するためにデータ・ハンドラーに渡されるメタオブジェクトです。このメタオブジェクトが静的なメタオブジェクト内に指定されている場合は、これによって DataHandlerConfigMO コネクター・プロパティーで指定された値がオーバーライドされます。さまざまなビジネス・オブジェクト・タイプを処理するために異なるデータ・ハンドラーが必要な場合、この静的なメタオブジェクト・プロパティーを使用します。このプロパティーが動的な子メタオブジェクトで定義されている場合は、コネクター・プロパティーおよび静的なメタオブジェクト・プロパティーをオーバーライドします。データ・フォーマットが実際のビジネス・データに依存する可能性がある場合は、要求処理に動的な子メタオブジェクトを使用してください。指定したビジネス・オブジェクトは、コネクターによってサポートされている必要があります。
DataHandlerMimeType	特定の MIME タイプに基づくデータ・ハンドラーを要求することを許可します。このプロパティーが静的なメタオブジェクト内に指定されている場合は、これによって DataHandlerMimeType コネクター・プロパティーで指定された値がオーバーライドされます。さまざまなビジネス・オブジェクト・タイプを処理するために異なるデータ・ハンドラーが必要な場合、この静的なメタオブジェクト・プロパティーを使用します。このプロパティーが動的な子メタオブジェクトで定義されている場合は、コネクター・プロパティーおよび静的なメタオブジェクト・プロパティーをオーバーライドします。データ・フォーマットが実際のビジネス・データに依存する可能性がある場合は、要求処理に動的な子メタオブジェクトを使用してください。DataHandlerConfigMO で指定されたビジネス・オブジェクトには、このプロパティーの値に対応する属性が含まれている必要があります。

プロパティー名	説明
DoNotReportBusObj	必要に応じて、ユーザーは DoNotReportBusObj プロパティーを含めることができます。このプロパティーを true に設定すると、発行されるすべての PAN レポート・メッセージのメッセージ本体がブランクになります。要求側が、要求が正常に処理されたことは確認したいが、ビジネス・オブジェクトに対する変更の通知は必要としないという場合に、このプロパティーの設定をお勧めします。これはNAN レポートには影響しません。 このプロパティーが静的なメタオブジェクトで指定されていない場合、コネクターは、デフォルトの false に設定し、メッセージ・レポートにビジネス・オブジェクトを取り込みます。 **注:** このプロパティーは、同期要求に対してのみ使用可能です。
InputFormat	InputFormat は、所定のビジネス・オブジェクトに関連付けるメッセージ・フォーマットです。検索されたメッセージがこのフォーマットである場合、可能であればそのメッセージは所定のビジネス・オブジェクトに変換されます。このプロパティーは、内部的に DataEncoding プロパティーに関連付けられ、InputFormat ごとに DataEncoding を 1 つのみ指定します。デフォルトの変換プロパティーを使用してこのプロパティーを設定しないでください。値はビジネス・オブジェクトへの着信メッセージと突き合わせるために使用されます。
OutputFormat	OutputFormat は、所定のビジネス・オブジェクトから作成されたメッセージに対して設定されます。OutputFormat が指定されていない場合は、入力フォーマットが使用されます (使用可能な場合)。OutputFormat が動的な子メタオブジェクトで定義されている場合、静的なメタオブジェクトで定義されている値はオーバーライドされます。
InputQueue	コネクターが新規メッセージを検出するためにポーリングする入力キュー。複数 のInputQueues を構成するため、また状況に応じて、異なるデータ・ハンドラーを各キューにマップするために、コネクター固有プロパティーを使用することができます。
OutputQueue	OutputQueue は、所定のビジネス・オブジェクトから生成されたメッセージの送信先となる出力キューです。OutputQueue が動的な子メタオブジェクトで定義されている場合、静的なメタオブジェクトで定義されている値はオーバーライドされます。
ResponseTimeout	応答待機時にタイムアウトまで待ち合わせる時間 (ミリ秒単位) を指定します。この値が定義されていなかったり、値がゼロより小さい場合、コネクターは応答を待機せずに即時に SUCCESS を戻します。ResponseTimeout が動的な子メタオブジェクトで定義されている場合、静的なメタオブジェクトで定義されている値はオーバーライドされます。

プロパティー名	説明
TimeoutFatal	このプロパティーが定義され、値が True の場合、ResponseTimeout で指定されている時間内に応答を受け取らなければ、コネクターは APP_RESPONSE_TIMEOUT を戻します。応答メッセージを待機しているその他のスレッドはすべて、即時に InterChange Server に APP_RESPONSE_TIMEOUT を戻します。これにより、InterChange Server はコネクターを終了します。TimeoutFatal が動的な子メタオブジェクトで定義されている場合、静的なメタオブジェクトで定義されている値はオーバーライドされます。

さらに、Default という名前の予約済みのプロパティーをメタオブジェクト内で定義することができます。このプロパティーが存在する場合、そのアプリケーション固有の情報は、すべてのビジネス・オブジェクト変換プロパティーにデフォルト値を指定します。

以下にメタオブジェクトのサンプルを示します。

プロパティー名	アプリケーション固有のテキスト
Default	DataEncoding=Text:UnicodeLittle; OutputFormat=CUST_OUT; OutputQueue=QueueA;ResponseTimeout=10000; TimeoutFatal=False

アプリケーション固有の情報

アプリケーション固有の情報は、セミコロンで区切られた名前と値のペアの形式で構成されています。例えば、次のようにします。

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

データ・ハンドラーから InputQueues へのマッピング

静的メタオブジェクトのアプリケーション固有情報の InputQueue プロパティーを使用して、データ・ハンドラーと入力キューを関連付けることができます。この機能は、それぞれフォーマットと変換要件が異なる複数の取引先を処理するときに便利です。これを行うには、以下のステップを実行する必要があります。

1. コネクター固有のプロパティーを使用して、1 つ以上の入力キューを構成します。
2. 各入力キューにキュー・マネージャーを指定し、アプリケーション固有の情報に、データ・ハンドラー・クラス名や MIME タイプとともに入力キュー名を指定します。

例えば、静的メタオブジェクト内の以下の属性は、データ・ハンドラーと CompReceipts という名前の InputQueue を関連付けます。

```
[Attribute]
Name = Cust_Create
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
```

```
IsRequired = false
AppSpecificInfo = InputQueue=//queue.manager/CompReceipts;DataHandlerClassName=
com.crossworlds.DataHandlers.MQ.disposition_notification;DataHandlerMimeType=
message/
disposition_notification
IsRequiredServerBound = false
[End]
```

入力フォーマットの多重定義

メッセージを取得する場合、コネクターは通常、入力フォーマットを、ビジネス・オブジェクトと動詞の組み合わせに固有のフォーマットと突き合わせます。その上でコネクターは、ビジネス・オブジェクトの名前とメッセージの内容をデータ・ハンドラーに渡します。そのため、データ・ハンドラーは、メッセージの内容が、ユーザーが必要とするビジネス・オブジェクトに対応するかどうかを検証することができます。

ただし、複数のビジネス・オブジェクトで同じ入力フォーマットが定義されていると、コネクターは、データ・ハンドラーにデータを渡す前に、そのデータがどのビジネス・オブジェクトを表しているか判別できません。このような場合、コネクターはデータ・ハンドラーにメッセージの内容のみを渡してから、生成されるビジネス・オブジェクトに基づいて変換プロパティーを探します。したがって、データ・ハンドラーは、メッセージの内容のみに基づいてビジネス・オブジェクトを判別しなければなりません。

生成されたビジネス・オブジェクトで動詞が設定されていない場合、コネクターは、任意の動詞を使用して、このビジネス・オブジェクトに対して定義されている変換プロパティーを検索します。見つかった変換プロパティー・セットが 1 つだけの場合、コネクターは、その指定した動詞を割り当てます。複数のプロパティーが見つかった場合は、コネクターは動詞を見分けることができないので、メッセージを失敗させます。

メタオブジェクトのサンプル

以下に示す静的メタオブジェクトでは、動詞 Create、Update、Delete、および Retrieve を使用して Customer ビジネス・オブジェクトを変換するようにコネクターが構成されます。属性 `Default` がメタオブジェクト内で定義されていることに注意してください。コネクターは、次の属性の変換プロパティーを使用します。

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

これを、他のすべての変換プロパティーのデフォルト値とします。したがって、属性によって別の値が指定されるか、または動的な子メタオブジェクトによってオーバーライドされる場合を除き、コネクターはすべてのビジネス・オブジェクトをキュー `CustomerQueue1` に送出してから、応答メッセージを待機します。応答が 5000 ミリ秒以内に到着しない場合、コネクターは即時に終了します。

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

動詞 Update および Delete 付きの Customer オブジェクト: 入力フォーマット MODIFY は、多重定義されます。動詞 Update 付きのビジネス・オブジェクト Customer と動詞 Delete 付きのビジネス・オブジェクト Customer の両方に対して定義されます。取得したこのフォーマットのメッセージを正常に処理するためには、ビジネス・オブジェクト名 (おそらくは動詞も) をメッセージの内容に含めて、データ・ハンドラーが識別できるようにする必要があります。要求処理の操作のために、出力フォーマットが定義されていないため、コネクターは入力フォーマット MODIFY を使用するいずれかの動詞用のメッセージを送信します。

動詞 Retrieve 付きの Customer オブジェクト: 属性 Customer_Retrieve は、動詞 Retrieve 付きでタイプ Customer のビジネス・オブジェクトが、フォーマット Retrieve のメッセージとして送信されるように指定します。デフォルトの応答時間はオーバーライドされ、コネクターは、タイムアウトまでに 10000 ミリ秒待機することになります (応答を受信しなければ、やはり終了します)。

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

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

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

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

動的な子メタオブジェクト

静的なメタオブジェクトを通じて必要なメタデータを指定することが困難であるかまたは不可能な場合、コネクターは必要に応じて、ビジネス・オブジェクト・インスタンスごとに実行時に指定されたメタデータを受け入れることができます。

コネクターは、コネクターに渡されるトップレベルのビジネス・オブジェクトに子として追加されている動的なメタオブジェクトから、変換プロパティーを認識して読み取ります。コネクターを構成するために使用される静的なメタオブジェクトを通じて指定可能な変換プロパティーは、動的な子メタオブジェクトの属性値に複製されます。

動的な子メタオブジェクトのプロパティーは、静的なメタオブジェクトのプロパティーをオーバーライドするので、動的な子メタオブジェクトを指定する場合は、静的なメタオブジェクトを指定するコネクター・プロパティーを取り込む必要はありません。したがって、動的な子メタオブジェクトは、静的なメタオブジェクトとは切り離して使用することができ、また静的なメタオブジェクトを動的な子メタオブジェクトと切り離して使用することも可能です。

注: コネクターは、動的な子メタオブジェクトを使用して同期イベント・デリバリー時にコラボレーション名を提供する機能をサポートしていません。

この前のセクションの表および次の表に、ビジネス・オブジェクト Customer_Create の静的な子メタオブジェクトおよび動的な子メタオブジェクトのサンプルを示しています。アプリケーション固有の情報は、セミコロンで区切られる名前と値のペアのフォーマットで構成されることに注意してください。

プロパティー名	値
DataEncoding	Text:UnicodeLittle
DataHandlerMimeType*	text/delimited
OutputFormat	CUST_OUT
OutputQueue	QueueA
ResponseTimeout	10000
TimeoutFatal	False

*DataHandlerConfigMO がコネクター構成プロパティーまたは静的なメタオブジェクトのいずれかで指定されたことを前提としています。

コネクターは、受け取ったトップレベル・ビジネス・オブジェクトのアプリケーション固有の情報をチェックして、タグ cw_mo_conn が子メタオブジェクトを指定するかどうかを判別します。指定している場合は、動的な子メタオブジェクトの値によって、静的なメタオブジェクト内で指定された値がオーバーライドされます。

ポーリング時の動的な子メタオブジェクトへのデータの取り込み

ポーリング時に取得したメッセージの詳細情報をコラボレーションに提供するために、コネクターは動的なメタオブジェクトの特定の属性にデータを取り込みます (作成したビジネス・オブジェクトですでに定義されている場合)。

以下の表に、ポーリング時の動的な子メタオブジェクトの構造を示します。

プロパティー名	サンプル値
InputFormat	CUST_IN
InputQueue	MYInputQueue
OutputFormat	CxIgnore
OutputQueue	CxIgnore
ResponseTimeout	CxIgnore
TimeoutFatal	CxIgnore

この表に示すように、動的な子メタオブジェクトでは、追加属性 InputQueue を定義することができます。この属性には、所定のメッセージの取得元のキューの名前が入ります。このプロパティーが子メタオブジェクトで定義されていない場合は、データは取り込まれません。

シナリオ例:

- コネクターは、キュー MyInputQueue から、フォーマットが CUST_IN のメッセージを取得します。
- コネクターは、このメッセージを Customer ビジネス・オブジェクトに変換し、アプリケーション固有のテキストをチェックして、メタオブジェクトが定義されているかどうか判別します。

- メタオブジェクトが定義されている場合、コネクターはこのメタオブジェクトのインスタンスを作成し、それに応じて InputQueue 属性と InputFormat 属性にデータを取り込んでから、ビジネス・オブジェクトを利用可能なコラボレーションにパブリッシュします。

動的な子メタオブジェクトのサンプル

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

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

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

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

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

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

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]

[Verb]
Name = Retrieve
[End]

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

JMS ヘッダー、WebSphere MQ メッセージ・プロパティー、および動的な子メタオブジェクトの属性

動的なメタオブジェクトに属性を追加して、メッセージ・トランスポートについて、詳細な情報を取得し、細かい制御を行うことができます。このような属性の追加により、JMS プロパティーを変更し、(アダプター・プロパティーで指定されたデフォルトの ReplyToQueue を使用するのではなく) 要求ごとに ReplyToQueue を制御し、またメッセージ CorrelationID を再始動することが可能になります。このセクションでは、このような属性と、同期モードと非同期モードの両方でそれらがイベント通知と要求処理に及ぼす影響について説明します。

以下の属性 (JMS ヘッダーおよび WebSphere MQ ヘッダー・プロパティーを反映) は、動的なメタオブジェクト内で認識されます。

表 4. 動的メタオブジェクトのヘッダー属性

ヘッダー属性名	モード	対応する JMS ヘッダー
CorrelationID	読み取り/書き込み	JMSCorrelationID
ReplyToQueue	読み取り/書き込み	JMSReplyTo
DeliveryMode	読み取り/書き込み	JMSDeliveryMode
優先順位	読み取り/書き込み	JMSPriority
Destination	読み取り	JMSDestination

表 4. 動的メタオブジェクトのヘッダー属性 (続き)

ヘッダー属性名	モード	対応する JMS ヘッダー
Expiration	読み取り	JMSExpiration
MessageID	読み取り	JMSMessageID
Redelivered	読み取り	JMSRedelivered
TimeStamp	読み取り	JMSTimeStamp
Type	読み取り	JMSType
UserID	読み取り	JMSXUserID
AppID	読み取り	JMSXAppID
DeliveryCount	読み取り	JMSXDeliveryCount
GroupID	読み取り	JMSXGroupID
GroupSeq	読み取り	JMSXGroupSeq
JMSProperties	読み取り/書き込み	

読み取り専用の属性は、イベント通知時にメッセージ・ヘッダーから読み込まれ、動的なメタオブジェクトに書き込まれます。これらのプロパティーは、要求処理の際に応答メッセージが発行された場合にも、動的なメタオブジェクトにデータを取り込みます。読み取り/書き込み属性は、要求処理時に作成されたメッセージ・ヘッダーで設定されます。イベント通知の際には、読み取り/書き込み属性をメッセージ・ヘッダーから読み込み、動的メタオブジェクトに取り込みます。

これらの属性の解釈および使用方法について、以下のセクションで説明します。

注: 上述の属性はいずれも必須ではありません。ビジネス・プロセスに関係する動的なメタオブジェクトに、任意の属性を追加することができます。

JMS プロパティー: 動的メタオブジェクト内の他の属性とは異なり、JMSProperties は、単一カーディナリティーの子オブジェクトを定義する必要があります。この子オブジェクト内の属性はすべて、以下のように、JMS メッセージ・ヘッダーの変数部分に読み書きされる単一のプロパティーを定義する必要があります。

1. 属性の名前はセマンティック値を持たないようにします。
2. 属性のタイプは、JMS プロパティー・タイプにかかわりなく常に String でなければなりません。
3. 属性のアプリケーション固有の情報には、属性がマップされる JMS メッセージ・プロパティーの名前とフォーマットを定義する、名前と値の 2 つのペアが含まれていなければなりません。

以下の表に、JMSProperties オブジェクト内の属性用に定義する必要のあるアプリケーション固有情報プロパティーを示します。

表 5. JMS プロパティー属性のアプリケーション固有情報

Name	指定可能な値	コメント
Name	任意の有効な JMS プロパティー名	これは JMS プロパティーの名前です。ベンダーによっては、拡張機能を提供するために特定のプロパティーを予約しています。一般にユーザーは、このようなベンダー固有の機能にアクセスしようとする場合を除き、JMS で始まるカスタム・プロパティーを定義すべきではありません。
Type	String、Int、Boolean、Float、Double、Long、Short	これは JMS プロパティーのタイプです。JMS API は、JMS メッセージ内に値を設定するためのメソッド (setIntProperty、setLongProperty、setStringProperty など) をいくつか提供しています。ここで指定する JMS プロパティーのタイプは、そのようなメソッドの中から、メッセージ内のプロパティー値の設定に使用するメソッドを指定します。

次の図では、動的なメタオブジェクト内の属性 JMSProperties と、JMS メッセージ・ヘッダー内の ID、GID、RESPONSE、および RESPONSE_PERSIST の 4 つのプロパティーの定義を示します。属性のアプリケーション固有の情報は、各プロパティーの名前およびタイプを定義します。例えば、属性 ID は、タイプ String の JMS プロパティー ID にマップされます。

非同期イベント通知: イベント・ビジネス・オブジェクト内にヘッダー属性を含む動的なメタオブジェクトが存在する場合、コネクターは、メタオブジェクトにトランスポート関連のデータを取り込むだけでなく、以下のステップを実行します。

1. メタオブジェクトの CorrelationId 属性に、メッセージの JMSCorrelationID ヘッダー・フィールド内で指定された値を取り込みます。
2. メタオブジェクトの ReplyToQueue 属性に、メッセージの JMSReplyTo ヘッダー・フィールド内で指定されたキューを取り込みます。このヘッダー・フィールドはメッセージ内の Java オブジェクトによって示されるため、属性にはキューの名前 (多くの場合 URI) が取り込まれます。
3. メタオブジェクトの DeliveryMode 属性に、メッセージの JMSDeliveryMode ヘッダー・フィールド内で指定された値を取り込みます。
4. メタオブジェクトの Priority 属性に、メッセージの JMSPriority ヘッダー・フィールドを取り込みます。
5. メタオブジェクトの Destination 属性に、メッセージの JMSDestination ヘッダー・フィールドの名前を取り込みます。Destination はオブジェクトによって表されるため、この属性には Destination オブジェクトの名前が取り込まれます
6. メタオブジェクトの Expiration 属性に、メッセージの JMSExpiration ヘッダー・フィールドの値を取り込みます。

7. メタオブジェクトの MessageID 属性に、メッセージの JMSMessageID ヘッダー・フィールドの値を取り込みます。
8. メタオブジェクトの Redelivered 属性に、メッセージの JMSRedelivered ヘッダー・フィールドの値を取り込みます。
9. メタオブジェクトの TimeStamp 属性に、メッセージの JMSTimeStamp ヘッダー・フィールドの値を取り込みます。
10. メタオブジェクトの Type 属性に、メッセージの JMSType ヘッダー・フィールドの値を取り込みます。
11. メタオブジェクトの UserID 属性に、メッセージの JMSXUserID プロパティー・フィールドの値を取り込みます。
12. メタオブジェクトの AppID 属性に、メッセージの JMSXAppID プロパティー・フィールドの値を取り込みます。
13. メタオブジェクトの DeliveryCount 属性に、メッセージの JMSXDeliveryCount プロパティー・フィールドの値を取り込みます。
14. メタオブジェクトの GroupID 属性に、メッセージの JMSXGroupID プロパティー・フィールドの値を取り込みます。
15. メタオブジェクトの GroupSeq 属性に、メッセージの JMSXGroupSeq プロパティー・フィールドの値を取り込みます。
16. メタオブジェクトの JMSProperties 属性に定義されたオブジェクトを調べます。アダプターは、このオブジェクトの各属性に、メッセージ内の対応するプロパティーの値を取り込みます。メッセージ内に特定のプロパティーが定義されていない場合、アダプターは属性の値を CxBlank に設定します。

同期イベント通知: 同期イベント処理の場合、アダプターは、イベントを送付して、統合ブローカーからの応答を待機し、その後応答メッセージをアプリケーションに返送します。ビジネス・データへの変更は、戻される応答メッセージに反映されます。アダプターは、イベントを創出する前に、上述の非同期イベント通知の場合とまったく同様にして、動的なメタオブジェクトへデータを取り込みます。動的なメタオブジェクト内に設定された値は、以下に説明する、応答によって発行されるヘッダーに反映されます (動的なメタオブジェクトの他のすべての読み取り専用属性は無視されます)。

- **CorrelationID** 動的なメタオブジェクトに属性 CorrelationId が含まれる場合、この属性を発信元のアプリケーションが予期する値に設定する必要があります。アプリケーションは、CorrelationID を使用して、コネクターから戻されたメッセージを元の要求と突き合わせます。CorrelationID の値が予期しないものであるか、無効である場合には、問題が生じます。この属性を使用する前に、相関関係にある要求メッセージと応答メッセージをアプリケーションが処理する方法を確認しておくと役立ちます。同期要求で CorrelationID にデータを取り込むには、4 種類の方法があります。
  1. 値を変更しません。応答メッセージの CorrelationID は、要求メッセージの CorrelationID と同じものになります。これは、WebSphere MQ オプション MQRO_PASS_CORREL_ID に相当します。
  2. 値を CxIgnore に変更します。コネクターは、デフォルトで、要求のメッセージ ID を応答の CorrelationID にコピーします。これは、WebSphere MQ オプション MQRO_COPY_MSG_ID_TO_CORREL_ID に相当します。

3. 値を CxBlank に変更します。コネクターは、応答メッセージで CorrelationID を設定しません。
4. 値をカスタム値に変更します。そのためには、応答を処理するアプリケーションがカスタム値を認識する必要があります。

メタオブジェクト内で属性 CorrelationID を定義しない場合、コネクターは自動的に CorrelationID を処理します。

- **ReplyToQueue** 属性 ReplyToQueue に異なるキューを指定して、動的なメタオブジェクトを更新した場合、コネクターは指定したキューに応答メッセージを送信します。これは推奨されません。要求メッセージ内に特定の応答キューを設定するアプリケーションは、そのキューで応答を待機することが前提となっているため、コネクターが異なるキューに応答メッセージを送信すると、通信が妨げられる可能性があります。
- **JMS Properties** 更新されたビジネス・オブジェクトがコネクターに戻されたときに動的なメタオブジェクト内の JMS Properties 属性に設定された値が、応答メッセージ内に設定されます。

Asynchronous 要求処理: コネクターは、要求メッセージを発行する前に、動的なメタオブジェクトを使用して (存在する場合)、要求メッセージにデータを取り込みます。コネクターは、要求メッセージを送信する前に、以下に示すステップを実行します。

1. 動的なメタオブジェクト内に属性 CorrelationID が存在する場合、コネクターはアウトバウンド要求メッセージの CorrelationID をこの値に設定します。
2. 動的なメタオブジェクト内で属性 ReplyToQueue が指定されている場合、コネクターは要求メッセージを通じてこのキューを受け渡し、このキューで応答を待機します。これにより、コネクター構成プロパティーで指定した ReplyToQueue の値をオーバーライドすることができます。さらに、負数の ResponseTimeout を指定した場合 (コネクターが応答を待機しないことを指定)、コネクターが実際には応答を待機しないのにもかかわらず、ReplyToQueue が応答メッセージ内に設定されます。
3. 属性 DeliveryMode が 2 に設定されている場合、メッセージを永続的に送信します。DeliveryMode が 1 に設定されている場合は、メッセージは永続的に送信されません。その他の値を設定すると、コネクターが異常終了する可能性があります。メタオブジェクト内で DeliveryMode が指定されていない場合、JMS プロバイダーが永続性設定を指定します。
4. 属性 Priority が指定されている場合は、コネクターが発信要求内に値を設定します。Priority 属性は、0 から 9 の値を取ることができます。その他の値を設定すると、コネクターが終了する可能性があります。
5. 動的なメタオブジェクト内で属性 JMSProperties が指定されている場合、動的な子メタオブジェクト内で指定された対応する JMS プロパティーが、コネクターによって送信されるアウトバウンド・メッセージに設定されます。

注: 動的なメタオブジェクト内のヘッダー属性が未定義であるか、または CxIgnore を指定する場合、コネクターはデフォルトの設定に従います。

Synchronous 要求処理: コネクターは、要求メッセージを発行する前に、動的なメタオブジェクトを使用して (存在する場合) 要求メッセージにデータを取り込みます。動的なメタオブジェクトにヘッダー属性が含まれる場合、コネクターは、応答

メッセージ内の対応する新しい値をその属性に取り込みます。コネクターは、応答メッセージの受信後、メタオブジェクトにトランスポート関連のデータを取り込むだけでなく、以下のステップを実行します。

1. 動的なメタオブジェクト内に属性 CorrelationID が存在する場合、アダプターは、応答メッセージ内で指定された JMSCorrelationID を用いてこの属性を更新します。
2. 動的なメタオブジェクト内で属性 ReplyToQueue が定義されている場合、アダプターは、応答メッセージ内で指定された JMSReplyTo の名前を用いてこの属性を更新します。
3. 動的なメタオブジェクト内に属性 DeliveryMode が存在する場合、アダプターは、メッセージの JMSDeliveryMode ヘッダー・フィールドの値を用いてこの属性を更新します。
4. 動的なメタオブジェクト内に属性 Priority が存在する場合、アダプターは、メッセージの JMSPriority ヘッダー・フィールドの値を用いてこの属性を更新します。
5. 動的なメタオブジェクト内で属性 Destination が定義されている場合、アダプターは、応答メッセージ内で指定された JMSDestination の名前を用いてこの属性を更新します。
6. 動的なメタオブジェクト内に属性 Expiration が存在する場合、アダプターは、メッセージの JMSExpiration ヘッダー・フィールドの値を用いてこの属性を更新します。
7. 動的なメタオブジェクト内に属性 MessageID が存在する場合、アダプターは、メッセージの JMSMessageID ヘッダー・フィールドの値を用いてこの属性を更新します。
8. 動的なメタオブジェクト内に属性 Redelivered が存在する場合、アダプターは、メッセージの JMSRedelivered ヘッダー・フィールドの値を用いてこの属性を更新します。
9. 動的なメタオブジェクト内に属性 TimeStamp が存在する場合、アダプターは、メッセージの JMSTimeStamp ヘッダー・フィールドの値を用いてこの属性を更新します。
10. 動的なメタオブジェクト内に属性 Type が存在する場合、アダプターは、メッセージの JMSType ヘッダー・フィールドの値を用いてこの属性を更新します。
11. 動的なメタオブジェクト内に属性 UserID が存在する場合、アダプターは、メッセージの JMSXUserID ヘッダー・フィールドの値を用いてこの属性を更新します。
12. 動的なメタオブジェクト内に属性 AppID が存在する場合、アダプターは、メッセージの JMSXAppID プロパティー・フィールドの値を用いてこの属性を更新します
13. 動的なメタオブジェクト内に属性 DeliveryCount が存在する場合、アダプターは、メッセージの JMSXDeliveryCount ヘッダー・フィールドの値を用いてこの属性を更新します。
14. 動的なメタオブジェクト内に属性 GroupID が存在する場合、アダプターは、メッセージの JMSXGroupID ヘッダー・フィールドの値を用いてこの属性を更新します。

15. 動的なメタオブジェクト内に属性 GroupSeq が存在する場合、アダプターは、メッセージの JMSXGroupSeq ヘッダー・フィールドの値を用いてこの属性を更新します。
16. 動的なメタオブジェクト内で属性 JMSProperties が定義されている場合、アダプターは、応答メッセージ内で指定された値を用いて子オブジェクト内で定義されたプロパティーを更新します。子オブジェクト内で定義されたプロパティーがメッセージ内に存在しない場合、値は CxBlank に設定されます。

注: 動的なメタオブジェクトを使用して、要求メッセージ内に設定された CorrelationID を変更しても、アダプターが応答メッセージを識別する方法には影響しません。アダプターは、デフォルトでは、すべての応答メッセージの CorrelationID が、アダプターの送信する要求のメッセージ ID と等しいことを予期します。

エラー処理: JMS プロパティーをメッセージから読み取ることができないか、またはメッセージへ書き込むことができない場合、コネクターがエラーをログに記録し、要求またはイベントは失敗します。ユーザーの指定した ReplyToQueue が存在しないか、またはアクセスできない場合、コネクターはエラーをログに記録し、要求は失敗します。CorrelationID が無効であるか、または設定できない場合、コネクターはエラーをログに記録し、要求は失敗します。すべての場合において、ログに記録されたメッセージは、コネクター・メッセージ・ファイルからのものです。

始動ファイルの構成

SAP Exchange Infrastructure のコネクターを始動するには、始動ファイルを構成する必要があります。

Windows

Windows プラットフォーム用のコネクターの構成を実行するには、以下のようにして start_WebSphereMQ.bat ファイルを修正する必要があります。

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

UNIX

Unix プラットフォーム用のコネクターの構成を実行するには、以下のようにして start_WebSphereMQ.sh ファイルを修正する必要があります。

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

アダプターの始動

コネクターの始動と停止、およびコネクターの一時始動ログ・ファイルの詳細については、ご使用のプラットフォーム用の「システム・インストール・ガイド」の始動に関する章を参照してください。

エラー処理

コネクターによって生成されるエラー・メッセージはすべて、WebSphere MQConnector.txt という名前のメッセージ・ファイルに保管されます。(ファイルの名前は、LogFileName 標準コネクター構成プロパティーによって決定されます。) 各エラーには、エラー番号とそれに続くエラー・メッセージが含まれます。

```
Message number
Message text
```

コネクターは、個別のエラーを以下の各セクションで説明するように処理します。

アプリケーション・タイムアウト

以下の場合に、エラー・メッセージ「ABON_APPRESPONSETIMEOUT」が戻されます。

- メッセージの検索中は、コネクターは JMS サービス・プロバイダーへの接続を確立できません。
- コネクターはビジネス・オブジェクトをメッセージに正常に変換しましたが、接続が確立されていないため、発信キューにそれを渡すことができません。
- コネクターはメッセージを発行したが、変換プロパティー TimeoutFatal が True に設定されたビジネス・オブジェクトの応答を待機していてタイムアウトになった。
- コネクターは、APP_RESPONSE_TIMEOUT または UNABLE_TO_LOGIN に等しい戻りコードを持つ応答メッセージを受け取りました。

アンサブスクライブされたビジネス・オブジェクト

コネクターが、アンサブスクライブされたビジネス・オブジェクトに関連したメッセージを取得した場合、コネクターは UnsubscribedQueue プロパティーによって指定されたキューへメッセージを送信します。

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

gotApplEvent() によって NO_SUBSCRIPTION_FOUND コードが戻されたとき、コネクターはメッセージを UnsubscribedQueue プロパティーが指定するキューへ送信し、他のイベントの処理を続行します。

コネクターがアクティブでない

gotApplEvent() メソッドが CONNECTOR_NOT_ACTIVE コードを戻すと、pollForEvents() メソッドは APP_RESPONSE_TIMEOUT コードを戻し、イベントは InProgress キュー内に残ります。

データ・ハンドラーによる変換

データ・ハンドラーがメッセージをビジネス・オブジェクトに変換できなかった場合や (JMS プロバイダーではなく) ビジネス・オブジェクトに固有の処理エラーが発生した場合、メッセージは、ErrorQueue で指定されたキューに送信されます。ErrorQueue が定義されていない場合、エラーが原因で処理できないメッセージは廃棄されます。

データ・ハンドラーがビジネス・オブジェクトをメッセージに変換できない場合は、BON_FAIL が戻されます。

トレース

トレースは、コネクターの動作を詳細に追跡するために使用できるオプショナル・デバッグ機能です。トレース・メッセージは、デフォルトでは STDOUT に書き込まれます。トレース・メッセージの構成の詳細については、17 ページの『第 2 章 アダプターのインストールと構成』に記載されている『コネクター構成プロパティー』を参照してください。トレースを使用可能にして設定する方法などのトレースの詳細については、「*コネクター開発ガイド*」を参照してください。

次に、コネクター・トレース・メッセージに推奨する内容を示します。

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

レベル 1　このレベルは、処理された各ビジネス・オブジェクトについて主要な情報を提供するトレース・メッセージや、ポーリング・スレッドが入力キュー内で新規メッセージを検出したときにそれを記録するトレース・メッセージに使用されます。

レベル 2　このレベルは、gotApplEvent() または executeCollaboration() のいずれかから、ビジネス・オブジェクトが InterChange Server へポストされるごとにログに記録するトレース・メッセージに使用されます。

レベル 3　このレベルは、メッセージからビジネス・オブジェクトへの変換およびビジネス・オブジェクトからメッセージへの変換に関する情報を提供するトレース・メッセージや、出力キューへのメッセージの送達に関する情報を提供するトレース・メッセージに使用されます。

レベル 4　このレベルは、コネクターがいつ関数を開始または終了したかを識別するトレース・メッセージに使用されます。

レベル 5　このレベルは、コネクターの初期化を示すトレース・メッセージ、アプリケーション内で実行されるステートメントを示すトレース・メッセージ、メッセージがキューから取り出されたりキューに入れられたりしたときにそれを記録するトレース・メッセージ、ビジネス・オブジェクトのダンプを記録するトレース・メッセージなどに使用されます。

第 3 章 ビジネス・オブジェクトの作成と変更

- 『ビジネス・オブジェクトの作成』
- 『ビジネス・オブジェクトの生成』

この章では、コネクターによるビジネス・オブジェクトの処理方法と、コネクターの前提事項について説明します。この情報は、新規のビジネス・オブジェクトをインプリメントするためのガイドとして使用できます。

ビジネス・オブジェクトの作成

WebSphere Business Integration システムのビジネス・オブジェクトでは、メタデータは、ビジネス・オブジェクト定義に格納され、コネクターとアプリケーションの対話に利用されます。メタデータ主導型コネクターは、サポートする各ビジネス・オブジェクトを処理する際に、コネクター内にハードコーディングされた命令ではなく、ビジネス・オブジェクト定義にエンコードされたメタデータに基づいて処理を行います。

ビジネス・オブジェクトのメタデータには、ビジネス・オブジェクトの構造、属性プロパティーの設定、およびアプリケーション固有情報の内容が含まれています。アダプターはメタデータ主導型であるため、コネクター・コードを変更する必要なしに、新規または変更されたビジネス・オブジェクトを処理することができます。ただし、コネクターの構成済みデータ・ハンドラーでは、サポートされるビジネス・オブジェクトの構造、オブジェクト・カーディナリティー、アプリケーション固有の情報の形式、およびビジネス・オブジェクトのデータベース表記に関する前提事項が想定されます。そのため、SAP XI のビジネス・オブジェクトを作成または変更する場合は、その変更がコネクターの従うべき規則に準拠している必要があります。そうすれば、コネクターは新規または変更されたビジネス・オブジェクトを正しく処理できます。

アダプターのインストール後、ビジネス・オブジェクトを作成します。ビジネス・オブジェクトの構造に関する要件は、構成済みデータ・ハンドラーによって課せられる要件だけで、その他にはありません。コネクターが処理するビジネス・オブジェクトは、統合ブローカーで許可される名前であれば、任意の名前にすることができます。命名規則の詳細については、「*Naming CrossWorlds Components*」を参照してください。

ビジネス・オブジェクトの生成

ビジネス・オブジェクトの生成には 2 つの方法があります。このうち主に実施されるのは、インポートされた IDOC 構造からビジネス・オブジェクトを生成する方法です。もう一つは、SAP インターフェース・リポジトリー (IFR) の XML スキーマ定義からビジネス・オブジェクトを生成する方法です。

XML スキーマ (XSD) 定義からのビジネス・オブジェクト生成

XML Object Discovery Agent (ODA) は、W3C ガイドラインに準拠する任意のスキーマまたは文書型定義 (DTD) からビジネス・オブジェクトを生成します。この機能

の詳細については、「*XML ODA User Guide*」を参照してください。SAP XI アダプターのビジネス・オブジェクトを生成するには、以下の手順を実行します。

1. URL ロケーション、または SAP XI リポジトリーから、XML スキーマ定義をダウンロードします。
2. XML ODA を起動します。

注: XML スキーマ定義から生成されたビジネス・オブジェクトを使用するときは、SAP XI でインターフェース・マッピングを定義する必要があります。

IDoc 構造からのビジネス・オブジェクト生成

ビジネス・オブジェクトの生成方法として主に実施されるのは、既存の IDoc 構造からビジネス・オブジェクトをインポートする方法です。SAP XI アダプターのビジネス・オブジェクトを生成するには、以下の手順を実行します。

1. SAP XI リポジトリーを始動します。
2. 使用するソフトウェア・コンポーネントをダブルクリックします。(システム・ランドスケープ・ディレクトリー (SLD) 内のソフトウェア・コンポーネントの保守については、SAP XI 資料を参照してください。)
3. SAP システムの接続データを保守します。
4. ご使用のソフトウェア・インターフェースで、「インポートされたオブジェクト (Imported Objects)」を右マウス・ボタンでクリックします。
5. 「RFC/IDoc をインポート (Import RFC/IDoc)」を選択します。
6. インポートされた IDoc 構造をローカル XSD スキーマ・ファイルに保管します。
7. XML ODA でこの IDoc スキーマ・ファイルを使用して、ビジネス・オブジェクトを生成します。

IFR IDoc XML スキーマからのビジネス・オブジェクト生成

XML Object Discovery Agent (ODA) は、W3C ガイドラインに準拠する任意のスキーマまたは文書型定義 (DTD) からビジネス・オブジェクトを生成します。この機能の詳細については、「*XML ODA User Guide*」を参照してください。SAP XI アダプターのビジネス・オブジェクトを生成するには、以下の手順を実行します。

1. http://ifr.sap.com に接続します。
2. ナビゲーション・バーから「インターフェース・リポジトリーの入力 (Enter the Interface Repository)」を選択します。
3. 必要なオブジェクトをダウンロードし、それを .xsd ファイルとして保管します。
4. XML ODA を起動し、Business Object Designer を使用して ODA に接続し、.xsd ファイルをロードします。

注: IFR IDoc スキーマから生成されたビジネス・オブジェクトを使用するときは、SAP XI でインターフェース・マッピングを定義する必要があります。この場合は、インポートされた IDOC 構造に対する IFR スキーマのインターフェース・マッピングを定義してください。

ビジネス・オブジェクト・プロパティーの例

XML データ・ハンドラーを備えた SAP XI コネクターのビジネス・オブジェクト・プロパティーの例を以下に示します。

```
 [ReposCopy]Version = 3.0.0

[End]

[BusinessObjectDefinition]

Name = MATMAS01

Version = 3.0.0

AppSpecificInfo = elem_fd=unqualified;attr_fd=unqualified

    [Attribute]

    Name = XMLDeclaration

Type = String

    Cardinality = 1

    MaxLength = 255

    IsKey = false

IsForeignKey = false

    IsRequired = false

AppSpecificInfo = type=pi

    IsRequiredServerBound = false

    [End]

    [Attribute]

    Name = ROOT

    Type = MATMAS01_MATMAS01

ContainedObjectVersion = 3.0.0

Relationship = Containment

Cardinality = 1

    MaxLength = 255

IsKey = false

    IsForeignKey = false
```

```
    IsRequired = false
    AppSpecificInfo = elem_name=MATMAS01
IsRequiredServerBound = false
    [End]
    [Attribute]
    Name = ObjectEventId
    Type = String
Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
IsRequired = false
    IsRequiredServerBound = false
    [End]
    [Verb]
    Name = Create
    [End]
    [Verb]
Name = Delete
    [End]
    [Verb]
Name = Retrieve
    [End]
    [Verb]
    Name = Update
    [End]
```

第 4 章 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 コネクター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 Broker (WMQI)
- 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

アダプターのリリース 2.3 で追加された標準プロパティーと削除された標準プロパティーは、以下の通りです。

新規プロパティー

- RFH2Message Domain
- ListenerConcurrency
- RestartCount
- WsifSynchronousRequestTimeout

削除されたプロパティー

なし

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
- コネクター固有構成プロパティー

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

Connector Configurator の使用

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

注: Connector Configurator と System Manager は、Windows システム上でのみ動作します。コネクターを UNIX システム上で稼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。コネクターがスタンドアロン・モードで (System Manager から独立して) 稼動している場合、例えば WebSphere MQ Integrator Broker で稼動している場合などには、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、または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。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ます。また、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場合があります。

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		Delivery Transport は JMS です
AdminOutQueue	*有効な JMS キュー名*	*CONNECTORNAME*/ADMINOUTQUEUE		Delivery Transport は JMS です
AgentConnections	1 から 4	1	コンポーネント再始動	ICS: Delivery Transport は MQ または IDL です
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	必須値
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	値なし	コンポーネント再始動	
ContainerManagedEvents	値なし、または JMS	JMS		保証付きイベント・デリバリー
ControllerStoreAndForwardMode	true または false	True	動的	ICS のみ
ControllerTraceLevel	0 から 5	0	動的	ICS のみ
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	コンポーネント再始動	JMS トランスポートのみ
DeliveryTransport	MQ、IDL、または JMS	JMS	コンポーネント再始動	WAS または WMQI: JMS のみ
DuplicateEventElimination	True/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	

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
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	サーバー再始動	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	サーバー再始動	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		サーバー再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		サーバー再始動	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	ICS のみ
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	ICS のみ
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	ICS のみ
ListenerConcurrency	1 から 100	1	コンポーネント再始動	ICS のみ: Delivery Transport は MQ でなければならない
Locale	en_US、ja_JP、ko_KR、 zh_C、zh_T、fr_F、 de_D、 it_I、es_E、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	ICS のみ
MaxEventCapacity	1 から 2147483647	2147483647	動的	ICS: Repository Directory は <REMOTE> でなければならない
MessageFileName	*パス/ファイル名*	*Connectorname*.txt あるいは 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

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
OADAutoRestartAgent	`True` または `False`	`False`	動的	ICS のみ: Repository Directory は <REMOTE> でなければならない
OADMaxNumRetry	*正数*	`1000`	動的	ICS のみ: Repository Directory は <REMOTE> でなければならない
OADRetryTimeInterval	*正数 (単位: 分)*	`10`	動的	ICS のみ: Repository Directory は <REMOTE> でなければならない
PollEndTime	`HH:MM`	`HH:MM`	コンポーネント再始動	
PollFrequency	*正整数 (単位: ミリ秒)* `no` (ポーリングを使用不可にする) `key` (コネクターのコマンド・プロンプト・ウィンドウで文字 `p` が入力された場合にのみポーリングする)	`10000`	動的	
PollQuantity	1 から 500	1	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
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		コンポーネント再始動	ICS の場合は <REMOTE> に設定する。WMQI または WAS の場合は *<local directory>* 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	
ResponseQueue	*有効な JMS キュー名*	*CONNECTORNAME*`/RESPONSEQUEUE`	コンポーネント再始動	
RestartCount	0 から 100		動的	コネクターはポーリング・モードで稼動している必要がある
RestartRetryCount	`0` から `99`	`3`	動的	
RestartRetryInterval	*適切な正数 (単位: 分)*	`1`	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
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 である場合のみ
SourceQueue	*有効な WebSphere MQ 名*	*CONNECTORNAME*/SOURCEQUEUE	コンポーネント再始動	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	
SynchronousRequestTimeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	
SynchronousResponseQueue		*CONNECTORNAME*/SYNCHRONOUSRESPONSEQUEUE	コンポーネント再始動	
WireFormat	CwXML、CwBO	CwXML	コンポーネント再始動	WMQI および WAS の場合は CwXML。Repository Directory が <REMOTE> の場合は CwBO (ICS)。
WisfSynchronousRequest 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

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

WebSphere ICS でのみ使用されます。

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、WMQI、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 ASCII という値が使用されています。このプロパティーの値を ascii7 または ascii8 に設定している場合、ASCII またはサポートされる他のいずれかの値に設定して、コネクターを構成し直す必要があります。

重要: 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。

デフォルト値は ascii です。

ConcurrentEventTriggeredFlows

WebSphere ICS でのみ使用されます。

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 JMS です。また、値なしに設定することもできます。

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 = 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 (Data Handler)」タブの値のフィールドは、ContainerManagedEvents を JMS に設定した場合のみ表示されます。

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

WebSphere ICS でのみ使用されます。

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

WebSphere ICS でのみ使用されます。

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

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 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 プロパティーの値は MQ、IDL、または JMS が指定可能で、デフォルトは IDL になります。
- WMQI がブローカー・タイプの場合は、指定可能な値は JMS のみです。
- WAS がブローカー・タイプの場合は、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する場合は、
`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・サイドの) コネクター・コントローラーと (クライアント・サイドの) コネクターの両方を始動するのは困難な場合があります。ご使用のシステムのプロセス・ヒープ・サイズが 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> の場合のみ適用できます。

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> の場合のみ適用できます。

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> の場合のみ適用できます。

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

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> の場合のみ適用できます。

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・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

統合ブローカーが ICS で、Repository Directory が <REMOTE> の場合のみ有効です。

異常シャットダウンの後、Object Activation Daemon (OAD) がアプリケーション固有のコンポーネントの再始動を自動的に試行するかどうかを指定します。このプロパティーは自動再始動を要求します。

デフォルト値は false です。

OADMaxNumRetry

統合ブローカーが ICS で、Repository Directory が <REMOTE> の場合のみ有効です。

異常シャットダウンの後で OAD がアプリケーション固有のコンポーネント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。

デフォルト値は 1000 です。

OADRetryTimeInterval

統合ブローカーが ICS で、Repository Directory が <REMOTE> の場合のみ有効です。

異常シャットダウンの後で OAD がアプリケーション固有のコンポーネントの再始動を自動的に試行する間隔 (分単位) を指定します。アプリケーション固有のコンポーネントが指定された間隔以内に始動しない場合、OAD は、『OADMaxNumRetry』で指定された回数だけ試行を繰り返します。

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 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 WMQI または WAS の場合には、この値を *<local directory>* に設定する必要があります。

ResponseQueue

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 の場合は、サーバーが要求を送信し、JMS 応答キューの応答メッセージを待ちます。

RestartCount

設定されている数のイベントの処理が完了すると、コネクターが自動的にシャットダウンしてから再始動します。イベントの数を RestartCount に設定します。このプロパティーを有効にするには、コネクターがポーリング・モードで稼動している必要があります (PollFrequency を “p” に設定します)。

要求処理時に設定されたイベント数を超えると、コネクターがシャットダウンし、次回ポーリングするときに再始動します。

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。

デフォルト値は 1 です。

RHF2MessageDomain

WebSphere MQ Integrator Broker 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 WebSphere MQ Integrator Broker に送信するときに、コネクター・フレームワークにより、ドメイン名、および固定値 mrm を含む JMS ヘッダー情報が書き込まれます。この構成可能なドメイン名により、ユーザーは WebSphere MQ Integrator Broker 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だけ表示されます。

SourceQueue

JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、 62 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は SOURCEQUEUE です。

SynchronousRequestQueue

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。

SynchronousResponseQueue

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。

SynchronousRequestTimeout

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。

- 統合ブローカーが WMQI または WAS の場合には、設定値は CwXML です。
- 統合ブローカーが ICS であり、RepositoryDirectory の値が <REMOTE> の場合には、設定値はCwBO です。

WisfSynchronousRequest 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。

第 5 章 トラブルシューティング

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

始動時の問題

問題	考えられる解決策と説明
初期化中にコネクターが予期せずシャットダウンし、メッセージ「Exception in thread "main" java.lang.NoClassDefFoundError: javax/jms/JMSException...」が報告される。	コネクターが IBM WebSphere MQ Java クライアント・ライブラリーからファイル jms.jar を検出できません。start_connector.bat の変数 WebSphere MQ_JAVA_LIB が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認してください。
初期化中にコネクターが予期せずシャットダウンし、メッセージ「Exception in thread "main" java.lang.NoClassDefFoundError: com/ibm/mq/jms/MQConnectionFactory...」が報告される。	コネクターが IBM WebSphere MQ Java クライアント・ライブラリーからファイル com.ibm.mqjms.jar を検出できません。start_connector.bat の変数 WebSphere MQ_JAVA_LIB が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認してください。
初期化中にコネクターが予期せずシャットダウンし、メッセージ「Exception in thread "main" java.lang.NoClassDefFoundError: javax/naming/Referenceable...」が報告される。	コネクターが IBM WebSphere MQ Java クライアント・ライブラリーからファイル jndi.jar を検出できません。start_connector.bat の変数 WebSphere MQ_JAVA_LIB が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認してください。
初期化中にコネクターが予期せずシャットダウンし、例外「java.lang.UnsatisfiedLinkError: no mqjbnd01 in shared library path」が報告される。	コネクターが IBM WebSphere MQ Java クライアント・ライブラリーから必須のランタイム・ライブラリー (mqjbnd01.dll [NT] または libmqjbnd01.so [Solaris]) を検出できません。パスに IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーが含まれていることを確認してください。
コネクターが「MQJMS2005: failed to create MQQueueManager for ':'」を報告する。	プロパティー HostName、Channel、および Port の値を明示的に設定します。

イベント処理

問題	考えられる解決策と説明
コネクターがデリバリーするすべてのメッセージに MQRFH2 ヘッダーが付いている。	MQMD WebSphere MQ ヘッダー付きのメッセージのみをデリバリーするには、出力キュー URI の名前の後ろに ?targetClient=1 を付けます。例えば、メッセージをキュー queue://my.queue.manager/OUT に出力する場合は、URI を queue://my.queue.manager/OUT?targetClient=1 に変更します。詳細については、17 ページの『第 2 章 アダプターのインストールと構成』を参照してください。

問題	考えられる解決策と説明
コネクターはデリバリーのときに、コネクター・メタオブジェクトでフォーマットがどのように定義されているかにかかわらず、すべてのメッセージ・フォーマットを 8 文字に切り捨てている。	これは、コネクターの制限ではなく、WebSphere MQ MQMD メッセージ・ヘッダーの制限です。

付録 A. 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 Broker (WMQI)
- WebSphere Application Server (WAS)

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る場合があります。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 (76 ページの『スタンドアロン・モードでの Configurator の実行』を参照)。

新しいアダプターをインストールする度に、コネクターの**構成ファイル**を作成する必要があります。この構成ファイルでは次の項目が設定されます。

- コネクターの標準プロパティーとアプリケーション固有のプロパティーが設定される。
- サポートするビジネス・オブジェクトとメタオブジェクトが指定される。

- コネクターが実行時に使用するロギング値とトレース値が設定される。
- アダプターのメッセージング・ハンドラーおよびデータ・ハンドラーにより使用されるプロパティー値が設定される。
- 既存のコネクターのコネクター・プロパティーを変更できるようになる。

この構成ファイルを作成し、構成ファイルの設定値を変更するには、Connector Configurator を使用します。

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

標準プロパティーはすべてのコネクター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、78 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 (すべてのブローカー)
- System Manager から実行 (ICS および WAS のみ)

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。ただし、統合ブローカーとして WMQI を使用している場合には、Connector Configurator はスタンドアロン・モードでのみ実行できます。

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: 統合ブローカー」**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 接続、WMQI 接続、WAS 接続のいずれかを選択します。

ブローカーとして ICS または WAS を使用するときに ICS または WAS で使用する構成ファイルを作成する場合には、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方法が便利です (83 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator 実行

統合ブローカーとして ICS または WAS を使用している場合には、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: 統合ブローカー」**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 接続または WAS 接続のいずれかを選択します。

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

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 (Connector-Specific Property Template)」**とクリックします。
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 (Connector-Specific Property Template)」**ダイアログ・ボックスが表示されます。
   - **「新規テンプレート」、「名前」**

     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、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 (Find Name)」**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。Connector Configurator には、デフォルト選択として、なにもプロパティー定義が含まれていない、**「None」**という名前のテンプレートが用意されています。

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。

- **プロパティーを編集**

  用意されているボタンを使用して (または**「プロパティーを編集」**表示内で右マウス・ボタン・クリックして)、テンプレートに新規プロパティーを追加したり、既存のプロパティーの編集や削除を行ったり、既存のプロパティーに子プロパティーを追加したりします。

  子プロパティーとは、別のプロパティー (親プロパティー) の属性です。親プロパティーには、単純値、子プロパティー、またはそれら両方を取り込むことができ

ます。親プロパティーと子プロパティーは階層の関係にあります。これらのプロパティーから構成ファイルを作成すると、Connector Configurator は階層プロパティー・セットを、親プロパティーの左側にボックスで囲んだ正符号 (+) を付けて識別します。

- **プロパティー・タイプ**

  Boolean、String、Integer、または Time のいずれかのプロパティー・タイプを選択します。

- フラグ

  このプロパティーに適用する、**「標準のフラグ」** (IsRequired、IsDeprecated、IsOverridden) または**「カスタム・フラグ」**(ブール演算子の場合) を設定できます。

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

値の指定

「値」タブを使用すると、プロパティーの最大長、最大複数値、デフォルト値、または値の範囲を設定できます。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、変更を行ってください。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. アダプターの表示パネルの左隅にあるボックスを右マウス・ボタン・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」**パネルがリフレッシュされ、**「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、作成した以前の値が表示されます。

   「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・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」**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」**表示で強調表示された依存プロパティーで、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 (Finish)」**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。

新しい構成ファイルを作成

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

ブローカーを選択するには、以下のステップを実行します。

- Connector Configurator のホーム・メニューで、**「ファイル」>「新規」>「コネクター構成」**をクリックします。**「新規コネクター」**ダイアログ・ボックスが表示されます。
- **「統合ブローカー」**フィールドで、ICS 接続、WMQI 接続、WAS 接続のいずれかを選択します。

  注: WMQI を選択するには、Connector Configurator を System Manager からではなく「スタート」メニューから起動する必要があります。
- 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を入力します。

ICS または WAS のファイルを新規に作成するには、次のいずれかの操作を行います。

- 「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、そのテンプレートを使用して構成ファイルを作成できます。

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

     コネクターの名前を入力します。名前では大文字と小文字が区別されます。入力する名前は、「connector」という単語で終わり、システムにインストールされたコネクターのファイル名との一貫性をもつ一意の名前でなければなりません。例えば、コネクター・ファイル名が `PeopleSoft.jar` である場合は、`PeopleSoftConnector` と入力します。

     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、WMQI 接続、WAS 接続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 (Connector-Specific Property Template)」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」>「プロジェクトに保管」**をクリックします。プロジェクトに保管するには、ブローカーとして ICS または WAS を使用しており、かつ 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、そのファイルを構成ファイル (*.cfg ファイル) として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」**とクリックします。
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」**とクリックします。

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 86 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、ICS 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。

「標準のプロパティー」と**「コネクター固有プロパティー (Connector-Specific Properties)」**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」**フィールドは通知を行うもので、構成できません。このフィールドでは、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指定します。

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る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すると、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きに、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・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、プロパティーに子プロパティーを追加するときは**「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. 85 ページの『標準コネクター・プロパティーの設定』で説明したように、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

アプリケーション固有のプロパティーは、**「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、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、暗号化が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・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

ほとんどのコネクター・プロパティーは静的プロパティーであり、**更新メソッド**は「COmponent restart」です。変更内容を有効にするには、変更したコネクター構成ファイルを保管した後でコネクターを再始動する必要があります。

サポートされるビジネス・オブジェクト定義の指定

Connector Configurator の**「サポートされているビジネス・オブジェクト」**タブで、コネクターが使用するビジネス・オブジェクトを指定します。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。

サポートされているビジネス・オブジェクトを指定するには、指定するビジネス・オブジェクトとそのマップがシステム内に存在している必要があります。

- 統合ブローカーが ICS または WAS の場合、ビジネス・オブジェクト定義とマップ定義が System Manager プロジェクトに保管されている必要があります。
- 統合ブローカーが WMQI の場合、ビジネス・オブジェクト定義と MQ メッセージ・セット・ファイルが存在している必要があります。

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」**リストの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、**「プロジェクトに保管」**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・オブジェクトにエージェント・サポートがある場合、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、大半のアプリケーション API では**「緊急」**レベルはサポートされないため、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 WMQI の場合

MQ メッセージ・セット・ファイル (`*.set` ファイル) には、メッセージ・セット ID が記述されています。この ID は、コネクターがサポートするビジネス・オブジェクトを指定するときに、Connector Configurator で必要となります。MQ メッセージ・セット・ファイルの作成の詳細については、「*WebSphere MQ Integrator Broker 用インプリメンテーション・ガイド*」を参照してください。

ビジネス・オブジェクト定義をシステムに追加するごとに、Connector Configurator を使用して、コネクターがサポートするビジネス・オブジェクトを指定する必要があります。

重要: コネクターがメタオブジェクトを必要とする場合は、ビジネス・オブジェクトと同様の方法で、各メタオブジェクトのメッセージ・セット・ファイルを作成し、それらのファイルを Connector Configurator にロードする必要があります。

サポートされるビジネス・オブジェクトを指定するには、以下の手順を実行します。

1. **「サポートされるビジネス・オブジェクト」**タブを選択し、**「ロード」**をクリックします。**「メッセージ・セット ID ファイルを開く」**ダイアログ・ボックスが表示されます。
2. コネクターのメッセージ・セット・ファイルを格納したディレクトリーに移動し、1 つ以上の適切なメッセージ・セット・ファイル `*.set` を選択します。
3. **「次へ」**をクリックします。**「ビジネス・オブジェクト名」**フィールドには、`*.set` ファイル内に格納されているビジネス・オブジェクト名が表示されます。各ビジネス・オブジェクトの数値メッセージ・セット ID が、対応する「メッセージ・セット ID」フィールドにリストされます。
   メッセージ・セット ID を変更しないでください。これらの名前と数値 ID は、構成ファイルを保管するときに保管されます。
4. ビジネス・オブジェクトを構成に追加するときには、ビジネス・オブジェクトのメッセージ・セット・ファイルをロードする必要があります。構成内にすでに存在するビジネス・オブジェクト名を含むメッセージ・セットをロードしようとした場合、または重複するビジネス・オブジェクト名を含むメッセージ・セット・ファイルをロードしようとした場合、Connector Configurator は重複を検出し、**「結果のロード」**ダイアログ・ボックスを表示します。

このダイアログ・ボックスは、重複が存在する 1 つ以上のビジネス・オブジェクト名を表示します。表示された各重複名の**「メッセージ・セット ID」**フィールド内でクリックし、使用するメッセージ・セット ID を選択します。

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 を選択する場合、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。

独立型方式で作業している (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できるビジネス・オブジェクトのリストが示されます。このリストから目的のビジネス・オブジェクトを選択します。

関連付けられているマップ (ICS のみ)

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

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

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

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

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

  これらは、**「サポートされているビジネス・オブジェクト」**タブで指定した、このコネクターでサポートされるビジネス・オブジェクトです。「サポートされているビジネス・オブジェクト」タブで、サポートされるビジネス・オブジェクトを追加指定した場合、それらの内容は、「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、コネクターごとに、サポートされる各ビジネス・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 (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 にデプロイします。
  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することができます。
すべてのコネクターでこの機能がサポートされ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。

メッセージングの構成 (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このデータ・ハンドラーを使用できるわけではありません。

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

構成ファイルの保管

コネクターの構成が完了したら、コネクター構成ファイルを保管します。Connector Configurator では、構成中に選択したブローカー・モードで構成ファイルが保管されます。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。

ICS の場合:

- System Manager から、ICS ユーザー・プロジェクトに `*.con` 拡張子付きファイルとして保管します。

- System Manager から、指定したディレクトリーに *.con 拡張子付きファイルとして保管します。
- ファイルをローカル構成ファイルとして使用する場合には、スタンドアロン・モードで、*.cfg 拡張子が付いたファイルとしてディレクトリー・フォルダーに保管します。

WMQI の場合:

- 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。

WAS の場合:

- System Manager から、WAS ユーザー・プロジェクトに *.con 拡張子付きファイルとして保管します。
- System Manager から、指定したディレクトリーに *.con 拡張子付きファイルとして保管します。
- 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。

構成ファイルを作成し、そのプロパティーを設定した後に、ご使用のコネクターに対応した適切な場所にこの構成ファイルを配置する必要があります。

- 統合ブローカーとして ICS を使用している場合は、構成を System Manager プロジェクトに保管し、System Manager を使用してそのファイルを ICS にロードします。
- 統合ブローカーとして WMQI を使用している場合は、構成ファイルを適切な場所にコピーします。これは、ご使用のコネクターの始動ファイルに指定されている構成ファイルの場所と完全に同じ場所でなければなりません。
- 統合ブローカーとして WAS を使用している場合には、ファイルを WAS ユーザー・プロジェクトに保管します。**「ファイル」>「エクスポート」**を選択し、.wsdl ファイルを作成します。作成された .wsdl ファイルは WSAD-IE へインポートできます。
  また、構成ファイルを .jar ファイルとして指定のディレクトリーへエクスポートできます。

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*」
- WMQI:「*WebSphere MQ Integrator Broker 用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**「ブローカー・タイプ」**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、以下に太字で示される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 B. クイック・ステップ

この付録では、SAP XI 用アダプターを要求処理およびイベント処理用に構成するためのクイック・ステップを示します。

要求処理

SAP XI 用アダプターを要求処理用に構成するには、以下の手順を行います。

1. 既存の XI リポジトリー・インターフェース・オブジェクトの XSD 定義をファイルに保管するか、あるいは http://ifr.sap.com から XSD IDoc 定義をダウンロードします。
2. WebSphere Business Integration Adapter for SAP XI とともに納入された XML ODA を使用して、ビジネス・オブジェクトを生成します。
3. 生成されたオブジェクトをサポートするようにアダプターを構成します。
4. SAP XI が未構成の場合は、以下のように構成します。
   a. WebSphere Business Integration Adapter for SAP XI のソフトウェア・コンポーネントをシステム・ランドスケープ・ディレクトリー (SLD) で定義します。
   b. インバウンド JMS アダプターを構成します。インターフェースごとに異なるインバウンド JMS アダプターが必要です。
   c. ディレクトリーを保守します。
5. エンド・アプリケーションからの要求オブジェクトをテストします。あるいは、テスト要求を送信する場合は Test Connector を使用します。

イベント処理

SAP XI 用アダプターをイベント処理用に構成するには、以下の手順を行います。

1. 既存の XI リポジトリー・インターフェース・オブジェクトの XSD 定義をファイルに保管するか、あるいは http://ifr.sap.com から XSD IDoc 定義をダウンロードします。
2. WebSphere Business Integration Adapter for SAP XI とともに納入された XML ODA を使用して、ビジネス・オブジェクトを生成します。
3. WebSphere Business Integration Adapter for SAP XI を以下のように構成します。
   a. 生成されたオブジェクトのサポートを追加します。
   b. 必要に応じて、ビジネス・プロセス (コラボレーション) およびマッピングを定義します。
   c. テストのため、エンド・アプリケーションを模擬するように Test Connector を構成します。
4. SAP XI を以下のように構成します。

a. WebSphere Business Integration Adapter for SAP XI のソフトウェア・コンポーネントをシステム・ランドスケープ・ディレクトリー (SLD) で定義します。
b. アウトバウンド JMS アダプターを構成します。インターフェースごとに異なるアウトバウンド JMS アダプターが必要です。
c. ディレクトリーを保守します。

5. IDoc またはその他のタイプのメッセージを起動するように SAP アプリケーションを構成します。
6. SAP アプリケーションでイベントを起動して、エンドツーエンドの接続性をテストします。

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