IBM WebSphere Business Integration Adapters

アダプター実装ガイド (WebSphere Application Server)

IBM WebSphere Business Integration Adapters

アダプター実装ガイド (WebSphere Application Server)

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、 221 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere Business Integration Adapter Framework 2.4、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
Implementing Adapters with WebSphere Application Server

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第2刷 2004.3

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

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

目次

本書について

IBM(R) WebSphere (R) Business Integration Adapter ポートフォリオは、主要な e-business テクノロジー、エンタープライズ・アプリケーション、およびレガシー・システムとメインフレーム・システムに、統合接続性を提供します。このシステムには、ビジネス・プロセス統合用のコンポーネントのカスタマイズ、作成、管理を行なうためのツールとテンプレートが組み込まれています。

本書では、WebSphere Application Server を統合ブローカーとして使用して、WebSphere Business Integration Adapters をインストール、構成、展開、および管理する方法について説明します。

注: 本書で使用する図は、構造と概念を表すことを目的とした例にすぎず、特定のビジネス・インテグレーションのシナリオを記述したものではありません。

対象読者

本書は、WebSphere Business Integration Adapters を WebSphere Application Server と共に使用できるように実装または管理するカスタマーおよびコンサルタントを対象としています。また、読者がすでに WebSphere Application Server Enterprise および WebSphere Studio Application Developer Integration Edition の構成方法と管理方法について知識があり、WebSphere MQ のメッセージングとメッセージ・フローを十分に理解していることを前提としています。

関連文書

WebSphere Business Integration Adapters の展開と管理に必要な資料は、以下に挙げるいくつかの異なる製品のライブラリーにまたがっています。

- WebSphere Business Integration Adapters の各製品
- WebSphere Application Server
- WebSphere Studio Application Developer Integration Edition
- WebSphere MQ

関連資料に関する情報と入手方法を以下に示します。

WebSphere Business Integration Adapters の資料

この資料一式では、WebSphere Business Integration Adapters のすべてのインストール・システムに共通の機能とコンポーネントを説明します。また、個々のコンポーネントに関する参照資料も含まれています。

次の IBM Web サイト `http://www.ibm.com/websphere/integration/wbiadapters/infocenter` では、この資料をインストールすることも、オンライン上でそのまま閲覧することもできます。

このサイトでは、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を記載します。

WebSphere Application Server の資料

WebSphere Application Server および WebSphere Studio Application Developer Integration Edition に関する資料は、IBM の Web サイト (http://www.ibm.com/software/webservers/appserv/library および http://www.ibm.com/software/awdtools/studiointegration/library) で参照およびダウンロードできます。

System Manager の資料

WebSphere Business Integration Adapters の構成および管理アクティビティーの多くは、System Manager というグラフィカル・ユーザー・インターフェースを使用して実行できます。System Manager についての詳細は、以下の 2 つのガイドを参照してください。

- 「*IBM WebSphere InterChange Server システム管理ガイド*」
- 「IBM WebSphere InterChange Server インプリメンテーション・ガイド」

これらは以下の Web サイトから入手できます。

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

WebSphere MQ の資料

WebSphere MQ 資料は、WebSphere MQ メッセージ・フォーマットおよびプロトコルに関する情報を提供します。これらの資料は次の IBM Web サイトよりブラウズおよびダウンロードできます。
http://www.ibm.com/software/integration/mqfamily

表記上の規則

本書では、以下のような規則を使用しています。

Courier フォント	コマンド名、ファイル名、入力情報、またはシステムが画面に表示する情報など、リテラル値を表します。
太字	初出語を示します。
イタリック、*イタリック*	変数名または相互参照を示します。
青い文字	オンラインで表示したときにのみ見られる青の部分は、相互参照用のハイパーリンクです。青い文字ストリングをクリックすることにより、参照先オブジェクトにジャンプすることができます。
{ }	構文の記述行の場合、中括弧 {} で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
[ ]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプションのパラメーターです。
...	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、option[,...] は、複数のオプションをコンマで区切って指定できることを意味します。

< >	命名規則では、不等号括弧によって名前の個々の要素が囲まれ、各要素を区別します (例：<server_name><*connector_name*>tmp.log)。
/、¥	本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。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。製品のパス名はすべて、ご使用のシステムで製品がインストールされたディレクトリーからの相対パス名です。
ProductDir	製品のインストール先ディレクトリーを表します。
%*text*% および $*text*	パーセント (%) 符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* であり、UNIX の *text* 環境変数の値を示します。
Windows: **UNIX:** **AIX:** **Solaris:**	先頭にこれらの記述のいずれかがあるパラグラフは、オペレーティング・システムの違いをリストする注記です。

改訂の要約

本章では、各リリースの「*アダプター実装ガイド (WebSphere Application Server)*」に加えられた主な変更点について要約しています。

WebSphere Business Integration Adapter Framework バージョン 2.4.0 の新機能

2004 年 2 月

本書では以下に挙げる内容が更新されました。

- System Manager および Eclipse Workbench について説明する 169 ページの『付録 E. System Manager と Eclipse Workbench』が追加されました。
- 117 ページの『第 9 章 ビジネス・インテグレーション・システムの管理』および 163 ページの『付録 C. コネクター始動オプション』では、コネクターの開始および停止についての情報が更新されました。

2003 年 12 月

- WebSphere Application Server 5.0.2 のサポートを含む、現在のリリースで更新されたインストール情報
- UNIX プラットフォーム向けに追加されたインストール情報
- アダプターの始動および停止に関する情報への Adapter Monitor の使用についての説明の追加
- 付録『WebSphere Business Integration Adapters のアップグレード』の追加
- 付録『Visual Test Connector の使用』の更新
- アダプターの新規インストールおよびパッケージ化に対応した変更

WebSphere Business Integration Adapters バージョン 2.3.1

「*アダプター実装ガイド (WebSphere Application Server)*」は、このリリースで新規追加されました。

第 1 部 概要と概念

第 1 章 WebSphere Business Integration Adapters の概要

WebSphere Business Integration Adapters (WBIA) 製品は、モジュラー・コンポーネントと、アプリケーションに依存しないビジネス・ロジックを使用します。WebSphere Business Integration Adapters がサポートする**ビジネス・インテグレーション・システム**は分散型で柔軟性に優れているため、カスタマイズ・フィーチャーを使って、サイト固有のニーズとアプリケーション固有のニーズを満たすことができます。

この実装ガイドでは、WebSphere Business Integration Adapters と WebSphere Application Server を使用する WebSphere Business Integration システムの展開方法および管理方法について説明します。このガイドは、次の 2 つの部に分かれています。

- 第 1 部では、WebSphere Business Integration Adapters ポートフォリオの概念的な概要と主要コンポーネントを説明します。また、統合ブローカーとの対話によりビジネス・インテグレーション・システムを作成するまでのアダプターのプロセスも説明します。第 1 部に示す情報は、WebSphere Application Server を統合ブローカーとして使用する WebSphere Business Integration Adapters にも、その他の統合ブローカーと共に使用する WebSphere Business Integration Adapters にも当てはまります (ただし、WebSphere Application Server に固有の情報であることが明記されている場合を除きます)。
- 第 2 部では、WebSphere Application Server を統合ブローカーとして WebSphere Business Integration Adapters をインストール、構成、および管理する際に特に役立つ、作業本位の情報を提供します。第 2 部の内容は、アダプターを WebSphere Application Server および WebSphere Studio Application Developer, Integration Edition と共に使用する場合にのみ当てはまります。

本章では、ビジネス・インテグレーション・システムのアーキテクチャーを説明し、WebSphere Business Integration Adapter のコンポーネントについて紹介します。次のセクションが含まれています。

必要な資料についての注意

ビジネス・インテグレーション・アダプターの展開に必要なすべてのタスクを実行するには、本書と合わせて、WebSphere Business Integration Adapters ライブラリーにあるほかの資料を使用する必要があります。特に、以下の資料が必要となります。

- *ビジネス・オブジェクト開発ガイド*
- 配置するアダプター用のアダプター・ユーザー・ガイド

カスタム・アダプターを作成する場合には、次の資料も使用する必要があります。

- *コネクター開発ガイド (Java 用)*
- *コネクター開発ガイド (C++ 用)*

ライブラリーにある資料の完全リストについては、 vii ページの『WebSphere Business Integration Adapters の資料』を参照してください。

WebSphere Business Integration システムとは

e-business を効果的に実行するには、企業は次の 2 つの課題を解決しなければなりません。

- 商取引を行なうために業務情報を各種ソース間で移動する必要がある。
- 企業環境での多種多様なアプリケーション間で業務情報を処理し、発送する必要がある。

ビジネス・インテグレーション・システムは、これらの 2 つのニーズのいずれにも柔軟性と拡張性を示しながら対応します。

トップレベルでは、**WebSphere Business Integration システム**は、1 つの統合ブローカーおよび 1 組のアダプターで構成されており、この構成によって異種の業務アプリケーションは、整合された情報転送を介して、ビジネス・オブジェクトという形でデータ交換を行なうことができます。

図 1 に、WebSphere Business Integration システムの略図を示します。

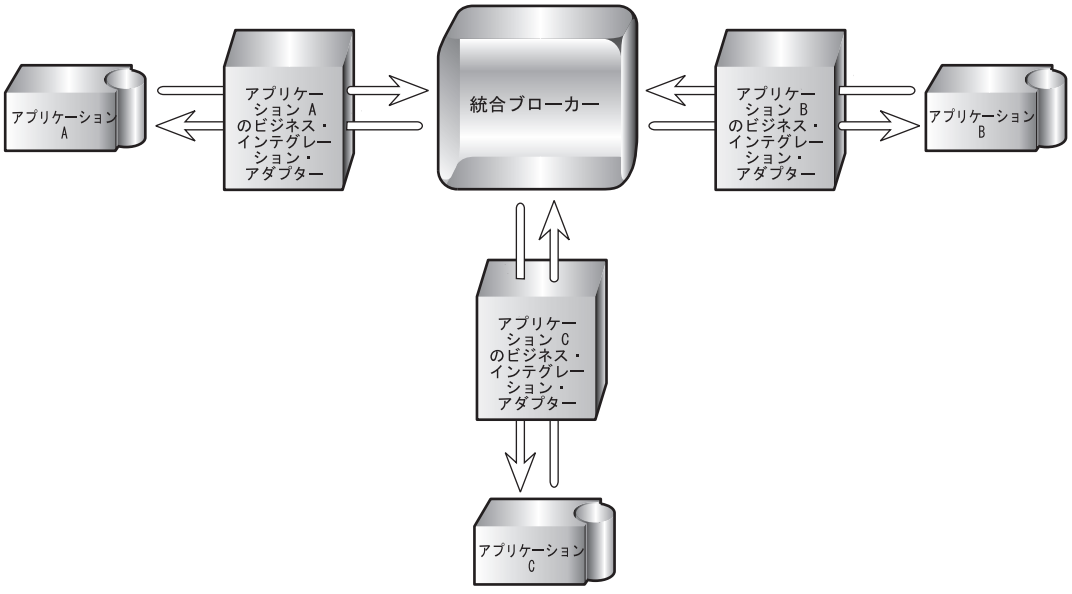


図 1. WebSphere Business Integration システムの上位図

統合ブローカーとは

統合ブローカーを使用すると、情報が適切な場所に正しい形式で到達するために必要な処理が実行され、異種アプリケーション間でそれぞれ異なる形式の情報を交換できるようになります。また、統合ブローカーを使用することによって、ユーザー定義の規則またはビジネス・ロジックを、データ処理に容易に適用できるようになります。データ交換は、統合ブローカーによって実行され、アプリケーション側は、データを受け取るアプリケーションのデータ規則や要件について何らの知識も必要としません。

WebSphere Business Integration Adapters とは

WebSphere Business Integration Adapters 製品は、一連のソフトウェア・プログラム、アプリケーション・プログラミング・インターフェース (API)、およびツールで構成されています。これらを使用して、統合ブローカーを介してアプリケーション間でビジネス・データを交換することができます。各ビジネス・アプリケーションをビジネス・インテグレーション・システムに組み込むには、アプリケーション固有のアダプターが必要です。

WebSphere Business Integration Adapters ポートフォリオの各アダプターには、次のものが組み込まれています。

- アプリケーションを統合ブローカーにリンクするコネクター
- コネクターを構成する際や、アプリケーションで必要なビジネス・オブジェクト定義を作成する際に役に立つグラフィカル・ユーザー・インターフェース付きのツール
- Object Discovery Agent (ODA)。アプリケーションのデータ・ストアに対して実行し、ビジネス・オブジェクト定義を作成します。ユーザーは、この後でこれを詳細化することができます。一部のアプリケーション用の WebSphere Business Integration Adapters には、ODA が組み込まれていませんので、ご注意ください。
- Object Discovery Agent Development Kit (ODK)。ODA を作成するために使用できる一組の API で構成されます。

特定のレガシー・アプリケーションまたは特化されたアプリケーション用に事前作成されたコネクターが IBM から提供されていない場合のために、カスタム・アダプターを開発するためのフレームワークを備えた、別売の Adapter Development Kit (ADK) が用意されています。

WebSphere Business Integration システムの機能

WebSphere Application Server が実装されている WebSphere Business Integration システムでは、アプリケーションと WebSphere Application Server の間でデータを移動するためのコネクティビティーは、WebSphere MQ キューを Java Message Service (JMS) プロバイダーとして使用する**コネクター**によって提供されます。コネクターは、どのマシンにも常駐することができ、そこから必要なキューへのアクセスやアプリケーションとの通信を行います。

各コネクターは、**コネクター・フレームワーク**と**アプリケーション固有のコンポーネント**の 2 つのパーツで構成されます。

- コネクター・フレームワークは、WebSphere MQ キューを使って統合ブローカーと対話します。
- アプリケーション固有のコンポーネントは、アプリケーションと直接対話します。

コネクターのサブコンポーネントを図 2 に示します。

図 2. コネクターのサブコンポーネント

データは、**アプリケーション固有のビジネス・オブジェクト**を使ってアプリケーション間で交換されます。これらのビジネス・オブジェクトは、コネクター・フレームワークと統合ブローカーの間で WebSphere MQ JMS メッセージ (ビジネス・オブジェクト・メッセージとも呼ばれる) として移送されます。

ビジネス・オブジェクトは、ビジネス・データをいくつかの目的でカプセル化および伝送します。ビジネス・オブジェクトは、次のものを搬送します。

- 新規データまたは変更データをソース・アプリケーションから宛先アプリケーションに搬送する。
- ソース・アプリケーションで作成したデータに対する要求を宛先アプリケーションに搬送する。
- データに対する要求に応答してアプリケーションによって戻されたデータを搬送する。

メタデータとしてエンコードされた、各データに関連付けられている命令は、検索、作成、または更新の対象となるデータの、アプリケーション・データベース内

におけるロケーションを指定します。ビジネス・オブジェクトの新規インスタンスは、ビジネス・オブジェクトの属性、値、およびメタデータの構造および編成を指定した、テンプレート (**ビジネス・オブジェクト定義**と呼ばれる) に基づいてアプリケーション固有のコンポーネントによって作成されます。

ビジネス・オブジェクト定義内のアプリケーション固有の情報や他のメタデータは、アプリケーション固有のコンポーネントのアクションをガイドします。そのため、アプリケーション固有のコンポーネントの振る舞いは**メタデータ主導型である**と言えます。アプリケーション固有のコンポーネントには、それがサポートするビジネス・オブジェクトのタイプごとに、ハードコーディングされた命令はありません。したがって、メタデータ主導型であるアプリケーション固有のコンポーネントは柔軟性があります。対応するアプリケーション・データがコネクターのメタデータ構文によって正確に記述することができるかぎり、アプリケーション固有のコンポーネントは、再コード化または再コンパイルすることなく、新規のビジネス・オブジェクト定義を自動的にサポートします。

ビジネス・インテグレーション・システムのデータ・フロー

ビジネス・インテグレーション・システムでは、データ・フロー (アプリケーション間またはエンティティー間で送信されるデータの移動および処理) は、ローカル・ネットワーク上またはインターネット上で、アプリケーション間の非同期または同期のデータ交換として発生します。

アプリケーションは、そのデータ・ストア内の変更を通知するため、またはデータを入手するために、別のアプリケーションとデータを交換することが必要となる場合があります。

ビジネス・インテグレーション・システム内のデータ交換は、以下のステップで構成されます。

1. イベント通知
2. 統合ブローカー処理
3. 要求処理

上記の各ステップを次に詳細に説明します。

イベント通知

変更されたアプリケーション・データを統合ブローカーに送るプロセスは**イベント通知**と呼ばれます。ビジネス・インテグレーション・システムに加入するアプリケーションのほとんどは、構成処理中に変更されて、**イベント・ストア** (アプリケーションのデータ変更やデータ要求をロギングするテーブルなど) が組み込まれます。アプリケーションに新たに変更された共用データがあることやアプリケーションが別のアプリケーションの情報を必要としていることを検出するために、コネクター・フレームワークは定期的間隔で**ポーリング呼び出し**を開始します。ポーリング呼び出しは、アプリケーションのイベント・ストアに変更があるかどうかをチェックするようアプリケーション固有のコンポーネントに依頼します。

最後のポーリング呼び出し以降に変更があった場合には、アプリケーション固有のコンポーネントは、変更データまたはデータ要求を表すビジネス・オブジェクト定義が存在するかどうかを判別します。コネクターのローカル・リポジトリーに適切

なビジネス・オブジェクト定義が存在することは、この特定の変更または要求を別のアプリケーションに伝達する必要があることを示します。アプリケーション固有のコンポーネントは、アプリケーション・データをビジネス・オブジェクト形態でコネクター・フレームワークに送信します。これは、**イベント・デリバリー**と呼ばれます。アプリケーション・データの変更やデータの要求はイベントと見なされるためです。

図 3 に、コネクターおよびそのサポート・インフラストラクチャーを示します。ここで、アプリケーションのデータ・ストアに対する変更を検出し、アプリケーション固有のビジネス・オブジェクトを組み立てて、変更データを統合ブローカーに搬送します。

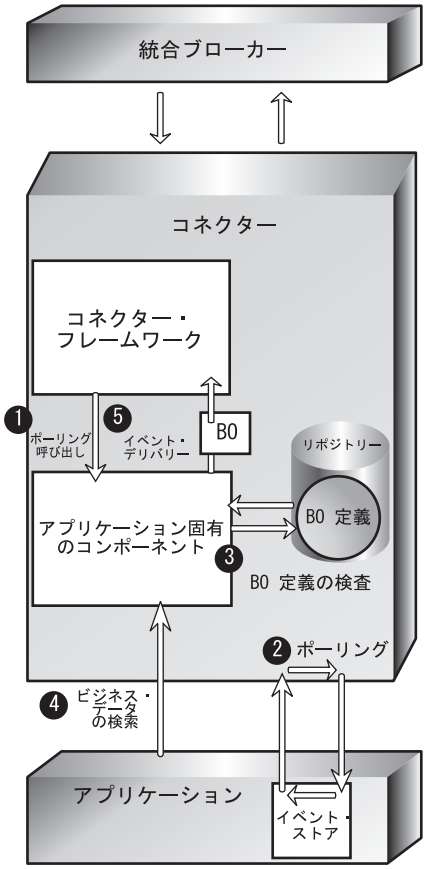


図 3. イベントを検出して引き渡すコネクター

図中の数値は、ステップの順序を示します。

1. コネクター・フレームワークは、アプリケーション固有のコンポーネントを開始して、アプリケーションのイベント・ストアに変更があるかどうかをチェックさせます。
2. アプリケーション固有のコンポーネントは、アプリケーションのイベント・ストアに変更があるかどうか調べるためにポーリングします。
3. アプリケーション固有のコンポーネントは、サポートされているビジネス・オブジェクト定義に変更データがマップされているかどうかを判定します。
4. アプリケーション固有のコンポーネントは、ビジネス・オブジェクトをインスタンス化し、それを使用して変更データを検索します。

5. アプリケーション固有のコンポーネントは、イベントの引き渡しを開始し、ビジネス・オブジェクトをコネクター・フレームワークに転送します。

ソース・アプリケーションのコネクター・フレームワークは、アプリケーション固有のビジネス・オブジェクトを受け取ると、WebSphere MQ キューに置くことが可能な WebSphere MQ メッセージに、そのビジネス・オブジェクトを変換して、統合ブローカーが受け取ることができるようにします。**データ・ハンドラー**は、ビジネス・オブジェクトを宛先 WebSphere MQ キューに応じた適切な XML ベースのワイヤー形式のメッセージに変換するために、コネクター・フレームワークが使用します。図 4 に、上記のプロセスを示します。

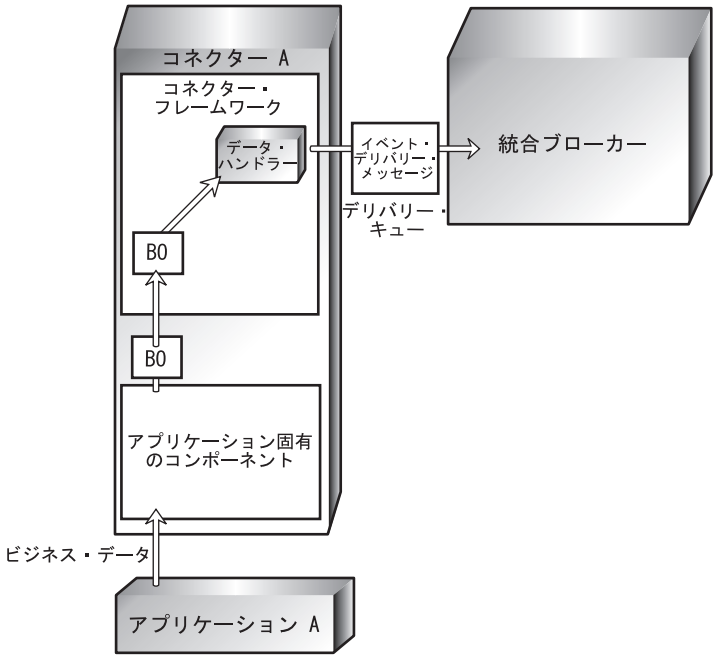


図 4. ビジネス・オブジェクトを MQ メッセージに変換するコネクター・フレームワーク

統合ブローカー処理

メッセージが統合ブローカーの WebSphere MQ イベント・デリバリー・キューに置かれると、統合ブローカーはそのメッセージを取り出して処理します。WebSphere Application Server 環境では、このタスクとそれに続く処理を実行するのは、WebSphere Studio Application Developer, Integration Edition を使用して作成された J2EE コンポーネントです。メッセージは、Message-Driven Bean (MDB) によってキューから取り出されて処理された後、EJB で処理されます。

統合ブローカーでの処理の結果、宛先アプリケーションに送信されるメッセージが生成されます。このメッセージは、WebSphere MQ 要求キューに置かれ、宛先アプリケーションのコネクター・フレームワークに転送されます。このメッセージを、**要求**と呼びます (WebSphere Application Server 環境ではハブ要求とも呼ばれます)。

要求処理

要求が宛先コネクターのキューに置かれると、**listen メカニズム**は、WebSphere MQ メッセージがその要求キューに到着したこと、および処理が必要であることを、宛先アプリケーションのアダプターのコネクター・フレームワークに通知します。コネクター・フレームワークは、データ・ハンドラーを起動して、宛先アプリケーションで処理可能なビジネス・オブジェクトに WebSphere MQ メッセージを変換します。図 5 を参照してください。

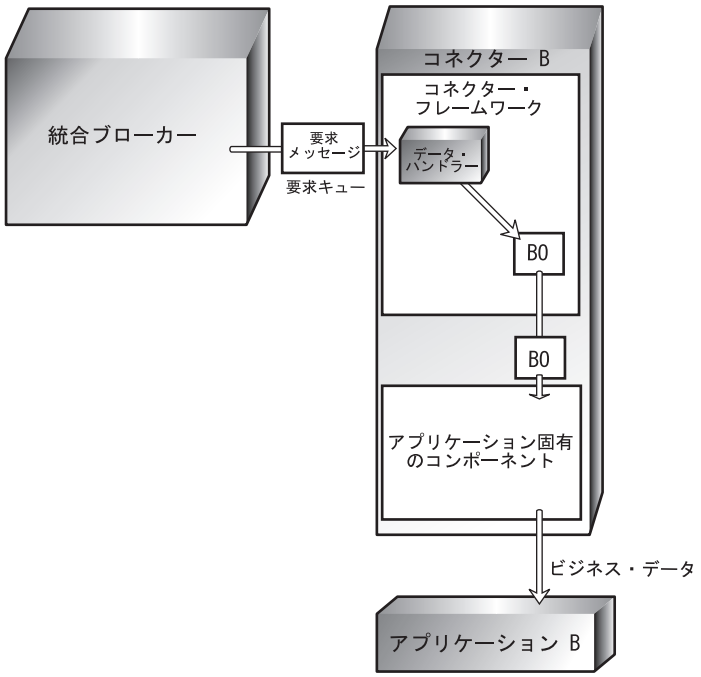


図 5. コネクターによる要求処理

場合によっては、要求が宛先アプリケーションからの**応答**を必要とすることがあります。一般に、応答は次のことを行なうために使用されます。

- ソース・アプリケーションが宛先アプリケーションに要求したデータを戻す
- ソース・アプリケーションが宛先アプリケーションに作成を依頼した新しいビジネス・エンティティー (カスタマーや注文など) についての情報をソース・アプリケーションに戻す

応答が必要な場合には、アプリケーション固有のコンポーネントは、情報を搬送するために要求ビジネス・オブジェクトを変更して、ビジネス・オブジェクトをコネクター・フレームワークに送り返します。コネクター・フレームワークは、データ・ハンドラーを呼び出してビジネス・オブジェクトを WebSphere MQ メッセージに変換し、発信元の要求メッセージで指定された応答先キューにメッセージを置きます。応答メッセージ内の相関 ID は、応答対象メッセージを表します。 11 ページの図 6 に応答処理が実行される仕組みを図示します。

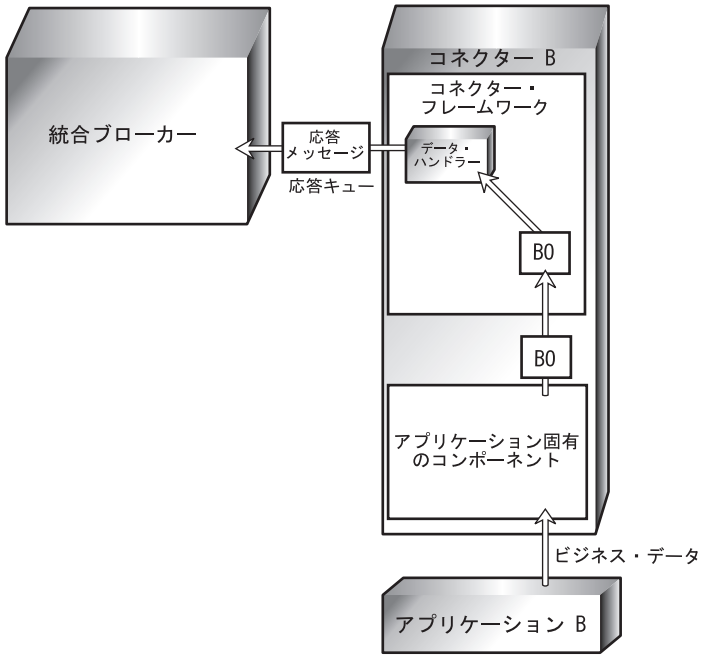


図 6. コネクターによる応答処理

ビジネス・インテグレーション処理の要約

これで、ビジネス・インテグレーション処理の各ステップが理解できたので、次にシステムを全体として復習します。ここでは、2 つのシナリオを示します。

変更データを別のアプリケーションに送信する例

ビジネス・インテグレーション・システムが実行するステップの一例を示します。このステップによってアプリケーション A は、アプリケーション B に同期をとるための変更データを送信することができます。

1. コネクター A のアプリケーション固有のコンポーネントはアプリケーション A にデータの変更があることを検出します。このコンポーネントは、この変更を伝達するためのビジネス・オブジェクト定義が存在すると判断し、そのビジネス・オブジェクト定義を使用して、変更情報を搬送するためのビジネス・オブジェクトを構成します。
2. アプリケーション固有のコンポーネントは、ビジネス・オブジェクトをコネクター・フレームワークに渡します。
3. コネクター・フレームワークはデータ・ハンドラーを起動し、ビジネス・オブジェクトを正しい XML ベースのワイヤー形式をもつ WebSphere MQ メッセージに変換して、そのメッセージを統合ブローカーの WebSphere MQ キューに置きます。
4. 統合ブローカーはメッセージを受け取り、処理します。これは、WebSphere Application Server 環境では、MDB がメッセージをキューから受け取り、適切な EJB に発送して処理させたときに該当します。

5. 統合ブローカーは、アプリケーション A のコネクターから発信されたメッセージの処理が完了すると、その結果生成されたメッセージを、アプリケーション B のコネクターの WebSphere MQ キューに置きます。
6. コネクター B のコネクター・フレームワークはメッセージをキューから取り出し、データ・ハンドラーを呼び出し、アプリケーション固有のコンポーネントによる処理が可能なビジネス・オブジェクトにメッセージを変換します。
7. アプリケーション B は、自分のカスタマー情報を更新してアドレスの変更を反映させます。

アプリケーション A がアプリケーション B にデータ変更を通知するのではなく、データを要求している場合には、アプリケーション B は、アプリケーション A に応答を返信する必要があります。次の例で、このシナリオを説明します。

別のアプリケーションからデータを入手する例

次に示すビジネス・インテグレーション・システムのステップでは、アプリケーション A は、あるカスタマーの最新の購入情報をアプリケーション B から検索します。

1. コネクター A のアプリケーション固有のコンポーネントは、アプリケーション A がアプリケーション B にデータを要求したことを検出します。このコンポーネントは、この要求を伝えるためのビジネス・オブジェクト定義が存在すると判断し、そのビジネス・オブジェクト定義を使用して、要求された情報を搬送するためのビジネス・オブジェクトを構成します。
2. アプリケーション固有のコンポーネントは、ビジネス・オブジェクトをコネクター・フレームワークに渡します。
3. コネクター・フレームワークはデータ・ハンドラーを起動し、ビジネス・オブジェクトを正しいワイヤー形式の WebSphere MQ メッセージに変換して、そのメッセージを統合ブローカーの WebSphere MQ キューに置きます。
4. WebSphere Application Server 環境では、統合ブローカー内の MDB がメッセージを受け取り、そのメッセージを処理する EJB を呼び出します。
5. 1 つ以上の EJB で着信メッセージが処理されます。その結果生成された発信メッセージは、アプリケーション B のコネクターの WebSphere MQ キューに置かれます。
6. コネクター B のコネクター・フレームワークはメッセージをキューから取り出し、データ・ハンドラーを呼び出し、アプリケーション固有のコンポーネントによる処理が可能なビジネス・オブジェクトにメッセージを変換します。
7. コネクター B のアプリケーション固有のコンポーネントは、要求で指定された情報を検索し、それをビジネス・オブジェクトとしてコネクター・フレームワークに渡します。
8. コネクター B のコネクター・フレームワークは、ビジネス・オブジェクトを応答メッセージに変換するためにデータ・ハンドラーを呼び出し、発信元の要求で指定された応答先キューにメッセージを置きます。

以降の章では、本章で紹介したビジネス・インテグレーション・コンポーネントについて記述するとともに、これらのコンポーネントの働きによってアプリケーションがデータを共用できるようになる過程についても詳細に説明します。

第 2 章 ビジネス・オブジェクト

ビジネス・オブジェクトとは、データ・エンティティー (1 つのまとまりとして扱われるデータの集合) を表します。データ・エンティティーの例としては、従業員に関する基本情報すべて (氏名、住所、電話番号、従業員番号、職位コード、月収など) を含む従業員レコードがあげられます。

ビジネス・インテグレーション・システムでは、エンティティーに含まれる情報を反映したビジネス・オブジェクトを作成します。本書では、データ・エンティティーは、それに含まれるビジネス情報の種類 (例えば、*従業員エンティティー* や*カスタマー・エンティティー* など) を指す意味でしばしば使われます。

ビジネス・オブジェクト定義とは、アプリケーション固有のコンポーネントがビジネス・オブジェクトの特定のインスタンスを作成するときのベースとなるテンプレートのことです。

本章では、ビジネス・オブジェクトについて詳細に紹介するとともに、ビジネス・インテグレーション・システムがビジネス・オブジェクトを使用して、アプリケーション間でデータを搬送する仕組みを説明します。本章は、次の各セクションから構成されています。

ビジネス・オブジェクトの役割

ビジネス・オブジェクトは、イベント、要求、または応答として機能できます。

イベント

ビジネス・オブジェクトには、**アプリケーション・イベント**が出現したことを報告する機能があります。これはアプリケーション内のデータ・エンティティーに影響を与えた操作が出現したことを意味します。アプリケーション・イベントとしては、そのデータ集合の値の作成、削除、または変更があげられます。

コネクターがアプリケーション・イベントを検出して、ビジネス・オブジェクトを統合ブローカーに送信する場合のビジネス・オブジェクトの役割は、イベントを表すことです。そのため、ビジネス・オブジェクトはビジネス・インテグレーション・システムで**イベント**と呼ばれます。

例えば、コネクターは、統合ブローカーに代わって、新しい従業員エンティティーがあるかどうかをチェックするためにアプリケーションをポーリングすることがあります。アプリケーションが新しい従業員エンティティーを作成した場合には、コネクターは統合ブローカーにイベント・ビジネス・オブジェクトを送信します。

要求

要求は一般に次のように生成されます。統合ブローカーは、コネクター・フレームワークにビジネス・オブジェクト・メッセージを**要求**として送信し、アプリケーション固有のコンポーネントを使用して、あるアプリケーション内のデータを挿入、変更、削除、または取得するように指示します。

応答

コネクターは、要求処理を完了すると、通常、統合ブローカーに**応答**を戻します。例えば、コネクターは、宛先アプリケーション内で従業員レコードを作成する要求を受け取ると、作成した従業員データと作成が正常に行われたことを示す状況表示とを添付してビジネス・オブジェクトを送信します。

ビジネス・オブジェクトの構造

ビジネス・オブジェクトは、タイプ (名前)、処理命令 (動詞)、およびデータ (属性値) が含まれている自己記述ユニットです。

図 7 は、単純なビジネス・オブジェクトの一例で、タイプ、動詞、および属性値が示されています。

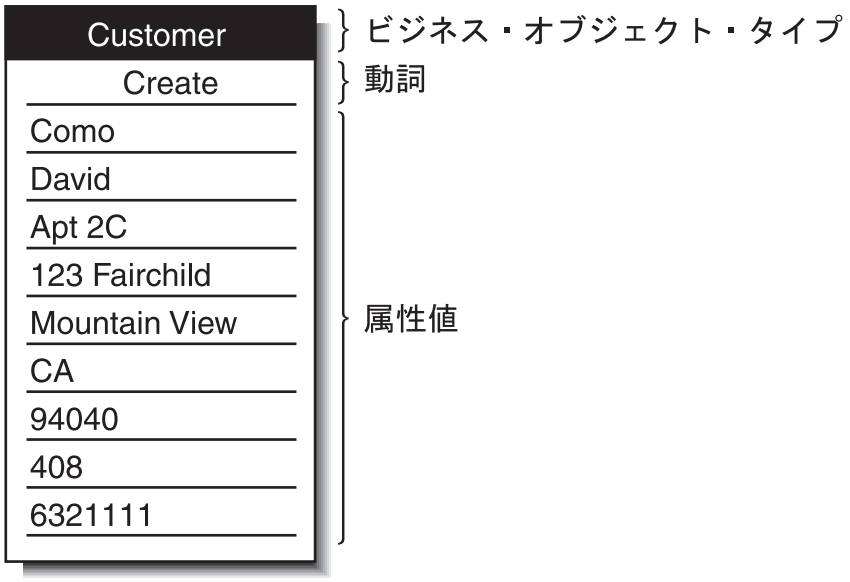


図 7. ビジネス・オブジェクトのコンポーネント

次のセクションでは、これらのコンポーネントについて説明します。

ビジネス・オブジェクト・タイプ

各ビジネス・オブジェクトには、ビジネス・インテグレーション・システムでそれぞれを識別するタイプ名が付いています。例えば、タイプには、Customer、Employee、Item、Contract などがあります。

ビジネス・オブジェクト動詞

ビジネス・オブジェクト動詞は、属性値に対応するアクションを指定します。動詞は、ビジネス・オブジェクトの役割によって、さまざまなタイプのアクションを指示することができます。表 1 では、3 つのビジネス・オブジェクトの役割をリストし、それぞれの役割を持つビジネス・オブジェクトでの動詞の意味について説明します。

表 1. ビジネス・オブジェクトの動詞の意味

ビジネス・オブジェクトの役割	動詞の意味
イベント	アプリケーションで起こったことを記述します。例えば、イベントで、動詞 Create は、ソース・アプリケーションが新しいデータ・エンティティーを作成したことを示します。
要求	ビジネス・オブジェクトを処理するためにアプリケーションと対話する方法をコネクターに指示します。例えば、動詞 Update は、コネクターに対するデータ・エンティティーの更新要求です。
応答	関連した要求で指定されている動詞をリストします。例えば、応答で動詞 Retrieve は、コネクターがアプリケーションから属性値を取得したことを示します。

注: IBM の表記規則では、*ビジネス・オブジェクト・タイプ.動詞* の形式を使用して、特定の動詞が指定された特定タイプのビジネス・オブジェクトを指示します。例えば、Customer.Create は、動詞 Create が指定されたビジネス・オブジェクト Customer です。

ビジネス・オブジェクト属性値

ビジネス・オブジェクトには、Last Name (ラストネーム)、First Name (ファーストネーム)、Employee ID (従業員 ID)、Invoice Status (送り状状況) など、データ・エンティティーに関連したデータ・フィールドを表す**属性値**が含まれます。

一部の属性には、データが含まれる代わりに、**子ビジネス・オブジェクト**または**子ビジネス・オブジェクトの配列**が含まれます。 16 ページの図 8 に、ビジネス・オブジェクト Contract (契約) の構造を示します。契約の Line Item (明細品目) 情報は、子ビジネス・オブジェクトの配列内にあります。

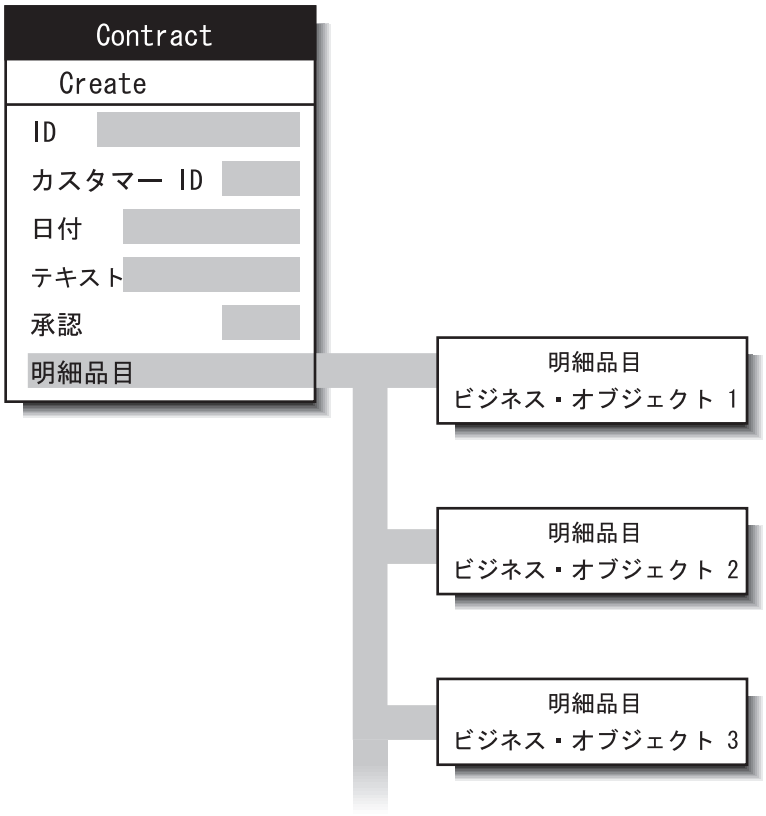


図 8. 子ビジネス・オブジェクトがあるビジネス・オブジェクト

子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を含んだビジネス・オブジェクトは、**階層ビジネス・オブジェクト**です。データのみを含んだ属性をもつビジネス・オブジェクトは、**フラット・ビジネス・オブジェクト**です。

ビジネス・オブジェクトのタイプ

ビジネス・オブジェクトには、アプリケーション固有のビジネス・オブジェクトと、汎用ビジネス・オブジェクトの 2 種類があります。アプリケーション固有のビジネス・オブジェクトは、どの統合ブローカーが使用されているかに関係なく、あらゆる WebSphere Business Integration システムで使用されます。汎用ビジネス・オブジェクトは、統合ブローカーが WebSphere InterChange Server (ICS) である場合に限り使用されます。WebSphere Application Server が統合ブローカーである場合には、アプリケーション固有のビジネス・オブジェクトのみが使用されます。したがって、本書でビジネス・オブジェクトというときはすべて、アプリケーション固有のビジネス・オブジェクトを意味しています。 IBM WebSphere Business Integration (WBI) Server の資料セットに含まれる多くの資料では、これら両方の環境が扱われるため、両タイプのビジネス・オブジェクトに言及しています。

- **アプリケーション固有のビジネス・オブジェクト**は、特定のアプリケーションまたは他のプログラマチック・エンティティーのデータ・エンティティー属性およびデータ・モデルを表します。
- **汎用ビジネス・オブジェクト**には、さまざまなアプリケーションに共通した、すなわち、特定のアプリケーションのデータ・モデルに限定されない、ビジネス関連属性のセットが含まれます。汎用ビジネス・オブジェクトは WebSphere

Application Server が統合ブローカーである場合には使用されません。ただし、WebSphere Business Integration Adapters ライブラリーと WebSphere InterChange Server ライブラリーの両方に含まれている資料では、汎用ビジネス・オブジェクトについても解説しています。

アプリケーション固有のコンポーネントは、更新などのアプリケーション・イベントを検出すると、該当するデータ・エンティティーをアプリケーションから取得し、それをビジネス・オブジェクトに変換します。

注: ドキュメンテーションでは、Clarify_Contact や Oracle_Customer などの、アプリケーション名が名前に含まれるビジネス・オブジェクトに言及するときは、アプリケーション固有のビジネス・オブジェクトを指します。例えば、ビジネス・オブジェクト Clarify_Contact には、Clarify アプリケーションが契約に関して保管する一連の情報が含まれます。別のアプリケーションでは、契約エンティティーに保管される情報の内容、順序、形式、または名前が多少異なる場合があります。

アプリケーション固有のコンポーネントは、ビジネス・オブジェクトを作成すると、そのビジネス・オブジェクトをコネクター・フレームワークに送信します。コネクター・フレームワークは、データ・ハンドラーを呼び出して、そのビジネス・オブジェクトを WebSphere MQ メッセージに変換して、統合ブローカーにディスパッチできるようにします。

ビジネス・オブジェクト定義とビジネス・オブジェクト

3 ページの『第 1 章 WebSphere Business Integration Adapters の概要』では、ビジネス・オブジェクトについて紹介しましたが、ビジネス・オブジェクト定義とビジネス・オブジェクトのインスタンスの区別については簡単に触れた程度でした。ここでは、その区別についてより詳細に説明します。

- **ビジネス・オブジェクト定義**では、各エンティティーにある情報のタイプと順序、およびサポートされる動詞を指定します。コネクターのローカル・リポジトリーには、ビジネス・オブジェクト定義が保管されます。
- **ビジネス・オブジェクト**は、実データを含む、定義のインスタンスです。ビジネス・オブジェクトは、ランタイムに作成され、リポジトリーに保管されません。

18 ページの図 9 に、ビジネス・オブジェクト定義とビジネス・オブジェクトの関係を示します。

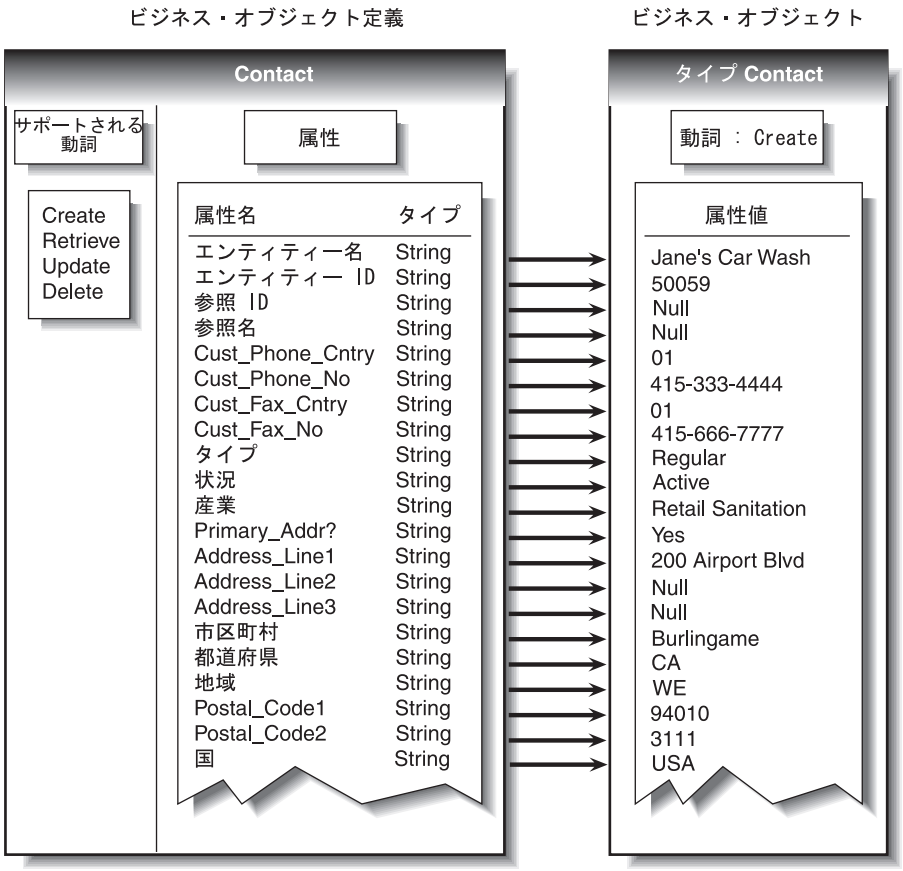


図 9. ビジネス・オブジェクト定義とビジネス・オブジェクト

ビジネス・オブジェクト定義のコンポーネント

一言で言うと、ビジネス・オブジェクトは、そのタイプ、その属性値、およびその動詞で表現されます。

全体として、ビジネス・オブジェクト定義はその名前で識別されます。名前は、Customer、VantiveCase、Invoice などの、ビジネス・オブジェクト定義タイプを示します。また、ビジネス・オブジェクトには、アプリケーション固有のコンポーネントがそれを処理するときに役に立つ、アプリケーション固有情報 (メタデータ) が含まれることもあります。また、すべてのビジネス・オブジェクトには、次の各セクションで説明する、属性および動詞も含まれます。

属性

ビジネス・オブジェクト定義にある**属性**には、Last Name (ラストネーム)、Employee ID (従業員 ID)、Case Number (事例番号)、Amount (量)、Date Initiated (開始日) などの、エンティティーと結びついた値が記述されます。実行時は、属性に実データが入力されます。

例えば、ビジネス・オブジェクト定義 Employee には従業員の氏名、住所、従業員 ID、および他の関連情報の属性が含まれています。ビジネス・オブジェクトの属性は、データベース表内のフォームまたは列のフィールドに類似しています。

また、属性は、契約の明細品目の配列、送り状のパーツ参照などの、子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を指すこともあります。

ObjectEventId 属性: ObjectEventId 属性は必須の属性で、いずれのビジネス・オブジェクト中でも最後におかれます。

コネクターは、イベントをパブリッシュすると、ビジネス・オブジェクト定義の ObjectEventId 属性を使用して、作成中のビジネス・オブジェクト・インスタンスを識別する固有値を保管します。一部のインプリメンテーションでは、ObjectEventId 属性の値がビジネス・インテグレーション・システムによって生成されます。さらにこの値は、システム内での特定のイベントの流れを識別および追跡するために使用されます。

基本属性タイプと複合属性タイプ: 属性のタイプが String、Boolean、Double、Float、Integer などの基本データ・タイプである場合、属性値は、データベース内のフィールドの値などの離散データです。例として、LastName、CustomerID、PartNumber、AssignedTo、Price などがあります。

属性のタイプが別のビジネス・オブジェクト定義の名前 (複合タイプ) である場合、属性値は子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列です。例として、Customer、Contract、Oracle_Contact などがあります。

属性のプロパティー: **プロパティー**の中には、属性が表す値を定義するものがたくさんあります。すべてではありませんが、図 10 に、ビジネス・オブジェクト定義での属性のプロパティーの位置を示します。

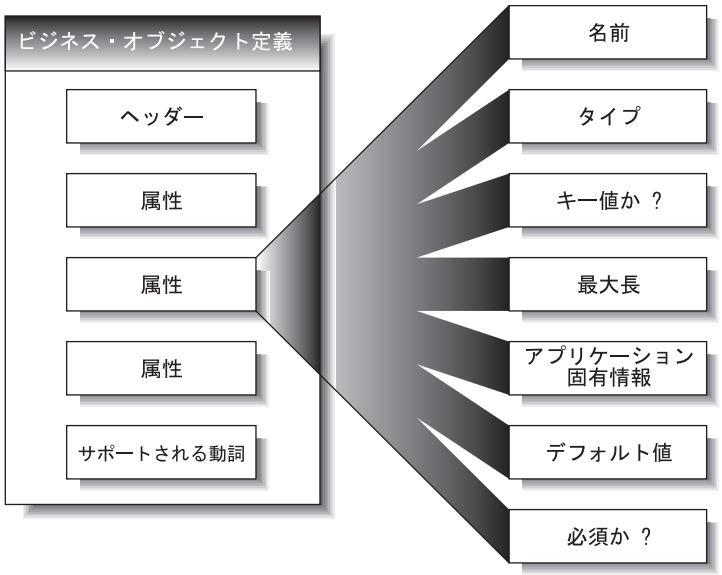


図 10. 属性のプロパティー

特定の属性のプロパティーのセットは、その属性タイプが基本タイプであるか、複合タイプであるかによって内容が異なります。すなわち、属性のプロパティーは、その属性が単一のデータ・ユニットを指すのか、子ビジネス・オブジェクトを指すのかによって異なります。

動詞

動詞は、ビジネス・オブジェクトのデータに対するアクションを示します。ビジネス・オブジェクト定義には、一連の動詞が含まれます。ビジネス・オブジェクトには 1 つの動詞のみが含まれます。

ビジネス・オブジェクト定義に関連した、最も一般的な動詞は Create、Retrieve、Update、および Delete です。

動詞の意味は、ビジネス・オブジェクトの役割によって異なります。動詞は、アプリケーション・イベントの記述、呼び出し、要求、または、前の要求の結果の確認を行なうことができます。

注: 一部のアプリケーションでは、ハード削除の要求はサポートされません。そのようなアプリケーションの場合、ビジネス・インテグレーション・システムは、同等の論理削除 (通常は、非アクティブ状態への更新) を行います。さらに、アプリケーションでハード削除がサポートされる場合であっても、そのアプリケーションに要求を送信する際に Delete 動詞を Update 動詞に変換するように、ビジネス・インテグレーション・システムを構成することができます。

ビジネス・オブジェクトの詳細

ビジネス・オブジェクトには、アプリケーション固有のコンポーネントが特定のアプリケーションに移動したり特定のアプリケーションから移動したりするデータが含まれます。したがって、各ビジネス・オブジェクト定義には、アプリケーションのデータ・モデルやアプリケーション固有のコンポーネントのアクセス・メソッドが反映されます。

2 つのアプリケーション固有のビジネス・オブジェクトが類似したアプリケーション・エンティティーを参照する場合でも、属性の編成手法や、それらのビジネス・オブジェクトのアプリケーション固有情報に違いが出ます。

属性の編成

アプリケーションが同じ情報を異なる手法で編成する場合がよくあります。例えば、アプリケーション A は連絡先の電話番号と FAX 番号を 4 つのフィールドに保管し、アプリケーション B は同じ番号を 2 つのフィールドに保管します。

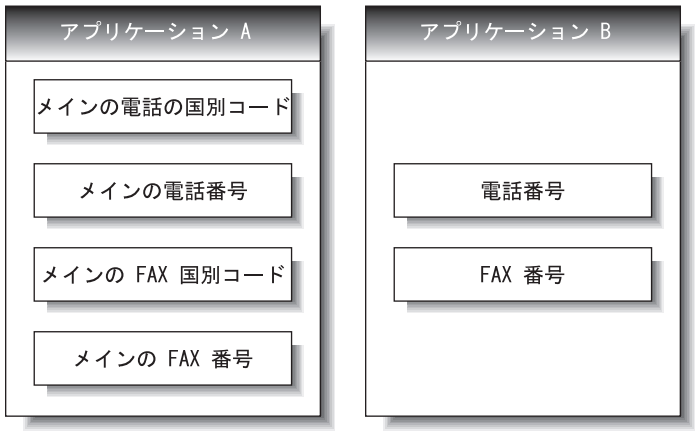


図 11. 2 つのアプリケーションでの電話データ

アプリケーション A およびアプリケーション B のビジネス・オブジェクト定義には、この違いを反映させるために異なる属性があります。

アプリケーション固有情報

また、ビジネス・オブジェクトが異なる理由として、それぞれのビジネス・オブジェクトには、オプションとして、そのアプリケーション固有のコンポーネントに対する組み込み処理命令を含めることができる点があげられます。これは、アプリケーション固有情報 (またはメタデータ) として参照されるもので、ビジネス・オブジェクトを処理するためにアプリケーション固有のコンポーネントが必要とする情報であればどんなものでも含めることができます。

ビジネス・オブジェクト定義には、ビジネス・オブジェクト全体、各属性、および各動詞それぞれに適用するアプリケーション固有情報を含めることができます。ビジネス・オブジェクト定義内でアプリケーション固有情報のある箇所それぞれに、アプリケーションとの対話でコネクターが使用する情報が示されます。

ビジネス・オブジェクトに関するアプリケーション固有情報

ビジネス・オブジェクトに関するアプリケーション固有情報は、ビジネス・オブジェクトを全体として処理する際にアプリケーション固有のコンポーネントが使用する情報を示します。

属性に関するアプリケーション固有情報

属性に適用するアプリケーション固有情報はしばしば、アプリケーション内の属性値ロケーションを識別します。アプリケーション固有のコンポーネントは、アプリケーションに対する API 呼び出しを作成する際にこの ID を使用して、属性値の取得または入力を行います。

アプリケーション固有情報は、アプリケーションごとに異なる形式をとります。アプリケーション固有のコンポーネントが、アプリケーションのフォーム名とフィールド名によって属性のロケーションを参照できる場合もあります。また、参照がより複雑な場合もあります。

22 ページの表 2 に、属性のアプリケーション固有情報に組み込むパラメーターの例を示します。これらのパラメーターは、データベース表内のデータを表すビジネ

ス・オブジェクトにのみ関するものです。

表 2. 属性アプリケーション固有情報の名前と値のペアのパラメーター例

パラメーター	説明
TN=*TableName*	データベース表の名前です。
CN=*col_name*	この属性のデータベース列の名前です。
FK=[..]fk_attributeName]	外部キー・プロパティーの値により親子関係が定義されます。
UID=AUTO	このパラメーターは、ビジネス・オブジェクト用の固有 ID を生成するように、またこの属性に値をロードするようにコネクターに通知します。
CA=*set_attr_name*	Copy Attribute (属性のコピー) プロパティーは、ある属性の値を別の属性にコピーするようコネクターに指示します。*set_attr_name* が現在の個々のビジネス・オブジェクト内の別の属性の名前に設定されている場合には、コネクターは、Create 操作でビジネス・オブジェクトをデータベースに追加する前に、指定された属性の値を使用して、この属性の値を設定します。
OB=[ASC|DESC]	Order By パラメーターの値が指定され、子ビジネス・オブジェクトに属性がある場合には、コネクターは検索クエリーの ORDER BY 文節内の属性の値を使用して、子ビジネス・オブジェクトを昇順に取得するか、降順に取得するかを判断します。
UNVL=*value*	コネクターが、ヌル値属性を持つビジネス・オブジェクトを取得するときにヌルを表すために使用する値を指定します。

1 つの属性アプリケーション固有情報が、上記にリスト例示したパラメーターをいくつか結合したものであることもあります。次の例では、パラメーターを区切るためにセミコロン (;) 区切り文字を使用しています。

```
TN=LineItems;CN=POid;FK=..PO_ID
```

上記の例では、アプリケーション固有情報が、表の名前、列の名前を指定しており、また現在の属性が子ビジネス・オブジェクトをその親にリンクする外部キーであることも指定しています。

例外的に、属性に関するアプリケーション固有情報が不要なケースもあります。例えば、アプリケーションによっては、データ・ユニットに関して非常に直接的で使用しやすい指定を行なう場合があります。あるアプリケーションが、表 3 に示すようなサンプル・フィールドを識別する場合を想定してください。

表 3. サンプル・アプリケーション ID

属性	値を持つフィールドのアプリケーション ID
CustomerID (カスタマー ID)	XCustomerID
CustomerName (カスタマー名)	XCustomerName
Status (状況)	XStatus
Industry (産業)	XIndustry

22 ページの表 3 に示されている例では、アプリケーション固有のコンポーネントは、容易に属性とアプリケーション内の ID を関連付けることができます。変換規則が、X の付加、または X の除去というように非常に規則的なためです。したがって、このアプリケーションに対するビジネス・オブジェクトの属性には、アプリケーション固有情報は不要であると思われます。

動詞に関するアプリケーション固有情報

ビジネス・オブジェクト定義には、サポートする各動詞に関するアプリケーション固有情報を含めることができます。アプリケーション固有情報は、その動詞がアクティブであるときにビジネス・オブジェクトの処理方法をアプリケーション固有のコンポーネントに指示します。

ビジネス・オブジェクト定義を作成または変更する方法

各コネクターには、他のアプリケーションに伝達するデータを定義する、ビジネス・オブジェクト定義のセットが必要です。アプリケーション固有のコンポーネントは、統合ブローカーにデータを送信する必要があるとき、それがサポートするビジネス・オブジェクト定義の 1 つから新しいビジネス・オブジェクトをインスタンス化します。コネクター構成プロセスのステップの 1 つに、サポートされるビジネス・オブジェクト定義を選択するステップがあります。ただし、まず最初に、アプリケーションに関するビジネス・オブジェクト定義を作成するか、または、別の方法で生成する必要があります。

ビジネス・オブジェクト定義の作成

アプリケーションに関するビジネス・オブジェクト定義を作成または取得する方法は、いくつかあります。

- アプリケーションに対応する Object Discovery Agent (ODA) が存在する場合には、それを使用してビジネス・オブジェクト定義を作成することができます。ODA は、アプリケーションの保管データの構造および編成を検証して、その検出結果に基づいてビジネス・オブジェクト定義を構成します。アプリケーションに対応する ODA が存在しない場合には、Object Discovery Agent 開発キット (ODK) を使用して ODA を作成します。
- Business Object Designer ツールを使用すると、ODA で生成されたビジネス・オブジェクト定義を変更するか、またはビジネス・オブジェクト定義を最初から組み立てて、ビジネス・オブジェクト定義を作成することができます。

その他の参考資料

「*ビジネス・オブジェクト開発ガイド*」には、ビジネス・オブジェクト定義の作成に関する詳細情報が記載されています。

さらに、多くのアダプターには、サンプル・ビジネス・オブジェクトが組み込まれています。組み込まれている場合には、サンプルは次のプロダクト・ディレクトリーにあります。

Windows:

¥connectors¥*ConnName*¥Samples

UNIX:

`/connectors/ConnName/Samples.`

ビジネス・オブジェクト定義の変更

追加のアプリケーション・データの取り込み、不要となったデータの収集停止、別のアプリケーションの変更への対応などの理由で、ビジネス・オブジェクト定義の変更が必要になる場合があります。「*ビジネス・オブジェクト開発ガイド*」で説明されている Business Object Designer ツールを使用すると、上記の変更を最も便利に行えます。

第 3 章 コネクター

コネクターは、ローカル・ネットワーク上でアプリケーションと統合ブローカーの間を仲介します。コネクターは、SAP R/3 バージョン 4 などの、アプリケーションに固有のものにすることも、XML や WebSphere MQ などの、データ・フォーマットやプロトコルに固有のものにすることもできます。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・フレームワークで構成されます。

共通の振る舞いについては、すべてのコネクターで共有されており、アプリケーションやビジネス・オブジェクトと対話する方法のみが異なります。本章では、コネクターの共通の振る舞いについて紹介するとともに、コネクターの異なる領域についても紹介します。本章は、次の各セクションから構成されています。

- 『コネクターの開始』
- 26 ページの『イベント通知』
- 33 ページの『要求処理』
- 32 ページの『保証付きイベント・デリバリー』
- 38 ページの『コネクター構成』
- 39 ページの『コネクターの開発』

一部の環境では、コネクターは「ブラック・ボックス」です。すなわち、その内部構成をあまり考慮せずに、コネクターを単にインストール、構成、管理、および使用することができます。ただし、カスタム・コネクターを作成する必要がある場合には、コネクターの振る舞いに関する詳細な知識が必要です。コネクターの作成・変更情報については、「*コネクター開発ガイド (Java 用)*」または「*コネクター開発ガイド (C++ 用)*」を参照してください。

コネクターの開始

コネクターは、開始スクリプトを使用して明示的に開始する必要があります。開始スクリプトは、コマンド行から、または Windows ショートカットから起動できます。

各コネクターには、コネクターの構成ファイルおよび各ビジネス・オブジェクト定義に関する個別の XML スキーマ文書を保持するローカル・リポジトリーがあります。このリポジトリーは、ローカル・ファイル・システム内の、アプリケーション固有のコンポーネントがインストールされるディレクトリーです。

開始時に、コネクターは次の作業を行います。

1. ローカル・リポジトリーからサポート・オブジェクト定義および構成プロパティーをロードする。
2. アプリケーションに接続する。

注: コネクターの開始時に統合ブローカーを実行している必要はありません。ただし、統合ブローカーがアクティブになるまで、データは転送されません。

イベント通知

注: “イベント通知” とは、コネクターのイベント通知メカニズムから発生するイベントの処理のことを指します。エージェント要求対話パターンでは、このようなイベントが記録されているメッセージの処理に、統合ブローカーからの応答が必要です。このことは、「同期イベント・デリバリー」または「同期イベント処理」と呼ばれる場合があります。エージェント・デリバリー対話パターンでは、統合ブローカーからの応答は必要ありません。このことは、「非同期要求処理」と呼ばれる場合があります。

トリガー・イベントをアプリケーションから受け取るコネクターは、それらのイベントに関する知識を取得し、統合ブローカーに関連データを送信する必要があります。図 12 に、イベント通知に関するコネクターの対話を示します。

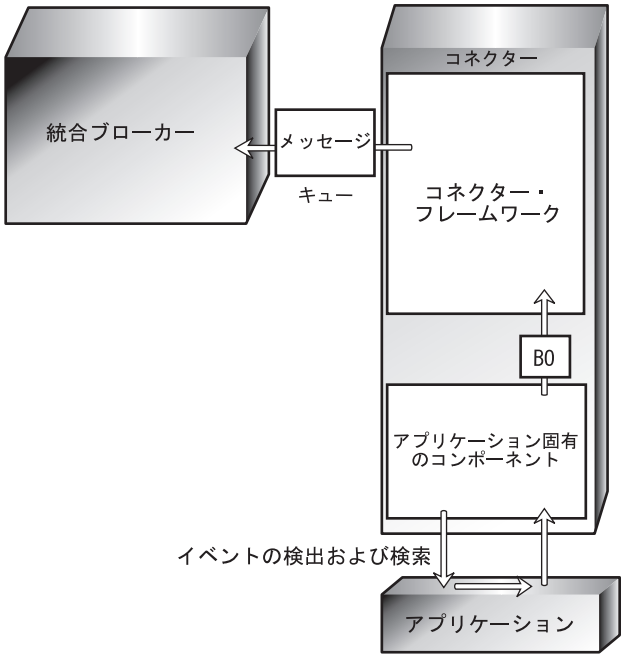


図 12. コネクター内のイベント通知

アプリケーション固有のコンポーネントがイベントを検出および取得する手法は、コネクターによって異なります。ただし、アプリケーション固有のコンポーネントがコネクター・フレームワークにイベントを送信する手法、およびコネクター・フレームワークが統合ブローカーにそれらのイベントを引き渡す手法は、すべてのコネクターで標準です。

次のサブセクションでは、次の項目を中心に、大半のコネクターの操作に関する一般的概念について説明します。

- コネクターが、アプリケーション・イベント通知メカニズムを使用する仕組み
- コネクターが、イベントを検出および処理する仕組み

ここでの説明は、特定のコネクターのインプリメンテーションを記述するものではありません。

アプリケーションのイベント通知メカニズムのセットアップ

コネクターにとって、アプリケーション・イベントとは、WebSphere Business Integration Adapters のビジネス・オブジェクト定義に関連付けられたアプリケーション・エンティティーのデータに影響を及ぼす何らかの操作のことです。アプリケーションには別のタイプのイベントがあります。例えば、マウス・クリックはアプリケーションのウィンドウ・システムまたはフォーム・インターフェースに対するイベントです。ただし、コネクターが関与するのは、事前定義されたデータ・レベル・イベントの一部であって、アプリケーションのデータ・ストアの内容を作成、更新、削除、その他の方法で影響するイベントのみです。

アプリケーションによっては、イベントを明示的にトラップして報告し、使いやすいイベント管理および構成可能なイベント・テキストを提供しているものがあります。個別的な報告可能イベントの概念がなく、なんらかの発生によりデータベースを更新しても、なにも通知しないことのあるアプリケーションもあります。WebSphere Business Integration Adapters は、両方のタイプのアプリケーションに対応したコネクターを備えています。

ほとんどのコネクターでは、コネクターの使用に関して**イベント通知メカニズム**をセットアップするために、アプリケーション構成が必要です。イベント通知メカニズムでは、アプリケーション内で発生する操作の番号付きリストを保持しています。このリストは、物理的には、アプリケーション・イベント・キュー、E メール受信箱、またはデータベース表の形式をとります。

コネクターが使用するイベント通知メカニズムには、どのようなタイプがあるのでしょうか。以降のセクションでは、一般的な方法をいくつか紹介します。

アプリケーションにイベント・サポートがある場合

アプリケーションがイベント・ベースの場合には、コネクターなど、クライアント・アプリケーションが使用するためのイベント通知インターフェースがおそらく用意されているはずです。また、アプリケーションでは、イベント・レポートのテキストを構成できる場合もあります。そのようなアプリケーションの場合、コネクターのイベント通知メカニズムのセットアップは、通常のアプリケーション・セットアップ・タスクです。

例えば、あるアプリケーションで、特定のタイプのイベントの発生時に実行するスクリプトをインストールでき、そのスクリプトがイベント受信箱に通知を入れることができる場合を想定します。このアプリケーションに対応したコネクターをインストールするには、そのコネクターのユーザー・アカウントを作成し、追跡するイベントを処理するスクリプトを作成または取得し、スクリプトをインストールし、各スクリプトを起動するイベントのタイプを指定し、受信箱を作成します。以上の作業を完了すると、アプリケーション固有のコンポーネントは、新しいイベントの発生をチェックするために受信箱の内容を定期的に検索します。

28 ページの図 13 に、イベント受信箱が組み込まれたアプリケーション構成を示します。

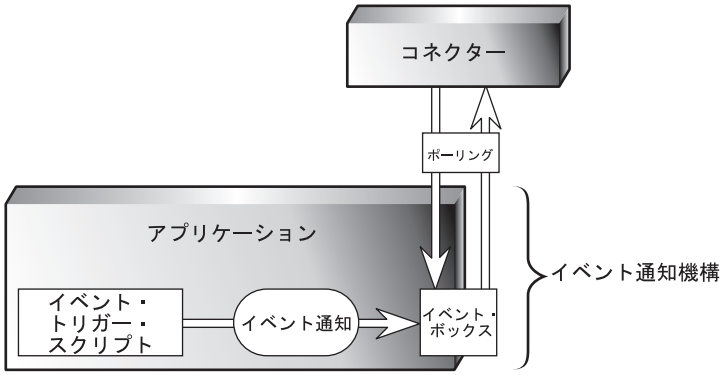


図 13. 例: イベント通知でのイベント受信箱の使用

特定の操作が実行されたときに E メール・メッセージの生成やイベント・キューへの書き込みができる内部ワークフロー・システムが用意されているアプリケーションもあります。 29 ページの図 14 に、ビジネス・オブジェクトおよびイベントが定義されている、独自のビジネス・オブジェクト・リポジトリーを持つアプリケーションを図示します。この図で、Customer はビジネス・オブジェクトで、Create、Delete、および Update は、このビジネス・オブジェクトに関連付けられているイベントのタイプです。

Customer.Update などのビジネス・オブジェクト・イベントが発生すると、イベントはワークフロー・システムに送信されます。続いてワークフロー・システムは、アプリケーション・データベース内のイベント表にエントリーを作成します。

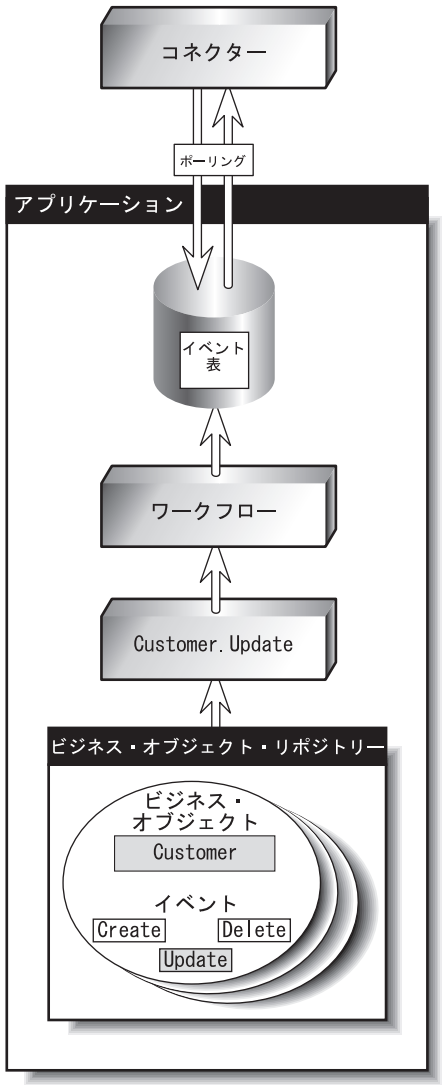


図 14. 例: イベント通知でのアプリケーション・ワークフローの使用

アプリケーションにイベント・サポートがない場合

コネクターがアプリケーション・イベントと対話するときの推奨メソッドは、アプリケーションの API を介する方法です。API には、アプリケーションのデータ・モデルとロジックを強制するフレームワークが用意されているからです。ただし、アプリケーション API によっては、イベント通知に対する固有のサポートを用意していないものもあります。

コネクターがそのようなアプリケーションからイベント通知を受け取ることができる方法の 1 つは、アプリケーション・データベースと対話することです。例えば、行に対する更新を検出するトリガーを、Employee 表にセットアップすることができます。更新が行われると、トリガーは更新に関する情報を、コネクターの展開時に作成した表に挿入します。イベント表に新たに表示される各行は、イベント通知を意味します。コネクターは SQL クエリーを使用して、表から新規のイベントを検索することができます。

図 15 に、上記の方法を図示します。

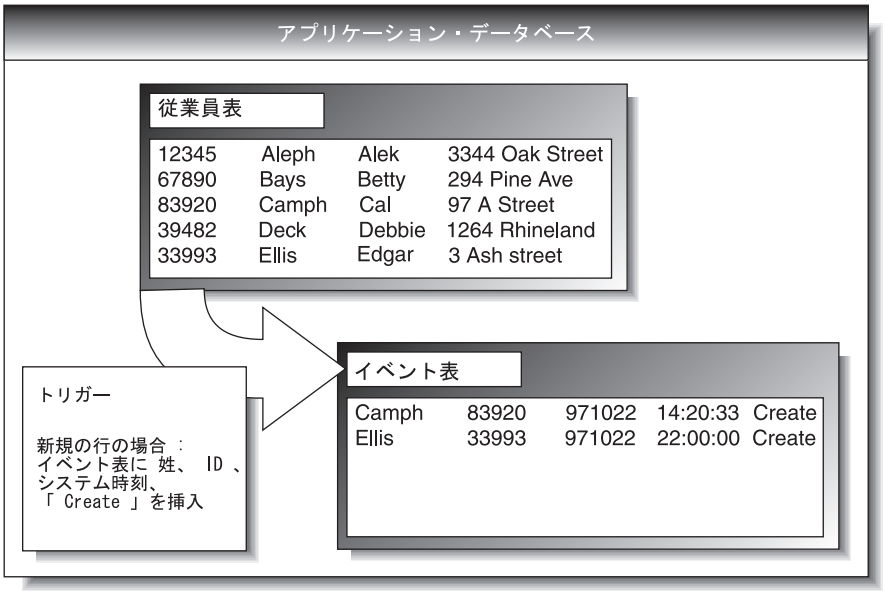


図 15. 例: イベント通知でのデータベースの使用

図 15 では、アプリケーション・データベースに、従業員表 (Employee Table) にレコードが作成される際のトリガーが用意されています。トリガーは、アプリケーションが新規のレコードを挿入するたびに、イベント表に 1 行を作成します。行には、新規従業員レコードのキー値 (ラストネームと従業員 ID)、システム時刻、およびイベント・タイプ Create が含まれます。

イベントの検出

コネクターのアプリケーション固有のコンポーネントは、そのイベント通知メカニズムを介してアプリケーション・イベントの情報を得ます。このメカニズムの最も一般的なものは、イベント・ストア内の新規イベントに対するポーリングです。ポーリング・メソッドは、コネクターが使用するイベント通知メカニズムに基づき、そのアプリケーションに固有のものです。

ポーリングは構成可能です。Connector Configurator ツールを使用してコネクターを構成する際に、次のことが行えます。

- アプリケーション固有のコンポーネントがアプリケーションをポーリングする頻度の調整
- アプリケーション固有のコンポーネントがアプリケーションをポーリングする時間帯の指定

大半のコネクターで、ポーリング呼び出しごとに処理するイベント数を指定することもできます。

アプリケーション固有のコンポーネントが、アプリケーションをポーリングする必要があるのは、別のアプリケーションがそのアプリケーションのイベントの情報を必要としている場合のみです。あるアプリケーションがイベントのソースではない場合は、Connector Configurator を使用して、ポーリング頻度を「no」に設定して、アプリケーション固有のコンポーネントのポーリングを停止することができます。

Connector Configurator の詳細については、「コネクター開発ガイド」、または配置するアダプター用のアダプター・ユーザー・ガイドを参照してください。

イベントの処理

イベントを検出すると、コネクターのアプリケーション固有のコンポーネントは次のことを行います。

- アプリケーション・イベントをビジネス・オブジェクト定義に関連付け、そのビジネス・オブジェクトのインスタンスを作成する
- ビジネス・オブジェクトに動詞およびキー値の属性を設定する
- アプリケーション・データを検索し、ビジネス・オブジェクトの属性を取り込む
- ビジネス・オブジェクトをコネクター・フレームワークに送信する
- イベントをアーカイブする (オプション)

アプリケーション・イベントをビジネス・オブジェクト定義と関連付ける方法

アプリケーション固有のコンポーネントは、イベントを検索するときに、そのイベントを表すビジネス・オブジェクト定義と動詞を判別する必要があります。

アプリケーション固有のコンポーネントは、イベント・テキストを使用して、ビジネス・オブジェクト定義と動詞にイベントを関連付けます。表 4 を参照してください。

表 4. イベント・テキストおよびビジネス・オブジェクトの形成

アプリケーション・イベント内のデータ・タイプ	例	用途
アプリケーション・エンティティー・タイプ	Customer、Part、Item	関連付けられたビジネス・オブジェクト定義を判別する
実行された操作	Create、Update、Delete	ビジネス・オブジェクトのアクティブな動詞を判別する

例えば、コネクターは次のイベント・テキストを Employee.Create ビジネス・オブジェクトに関連付けます。

```
1997.10.19.12:50.22 employee created lname="como" id="101961"
```

イベント・テキストは、次のもので構成されます (左から右へ)。

- イベントを一意的に識別するのに役に立つタイム・スタンプ
- アプリケーション・エンティティー「employee」
- 操作「created」
- 従業員のラストネームと従業員 ID、つまりアプリケーション固有のコンポーネントが残りの従業員情報を検索するのに使用する固有の ID (キー値)。

上記は簡単な例です。他のタイプのイベント・テキストでは、アプリケーション固有のコンポーネントによる処理をさらに必要とする場合があります。

アプリケーション固有のビジネス・オブジェクトの作成

イベントのビジネス・オブジェクト定義をサポートするようにコネクターが構成されている場合には、アプリケーション固有のコンポーネントはビジネス・オブジェクトを作成し、キー値を使用してアプリケーション・データを検索し、ビジネス・オブジェクトにアプリケーション・データを入力します。ビジネス・オブジェクトを作成するプロセスについては、35 ページの『ビジネス・オブジェクトの構成とデコンストラクション』を参照してください。

アプリケーション固有のビジネス・オブジェクトのコネクター・フレームワークへの送信

アプリケーション固有のコンポーネントは、ビジネス・オブジェクトをコネクター・フレームワークに送信します。このとき、このコンポーネントは、ビジネス・オブジェクトに格納されて搬送される情報の受信側となるビジネス・プロセスの ID を知っている必要はありません。

イベントのアーカイブ

アプリケーション・イベント・アーカイブは、トラブルシューティングおよびレコードの保持に役立ちます。イベント・アーカイブには、各イベントに関する、次に示すような状況情報が含まれます。

- 統合ブローカーに正常に送信されたこと
- 処理が失敗したこと

アプリケーションにイベント・アーカイブ・フィーチャーが用意されている場合には、コネクターは一般にそれを使用します。イベントのアーカイブをサポートしないアプリケーションのコネクターには、独自のイベント・アーカイブが備わっている場合があります。例えば、コネクターのイベント通知メカニズムが 30 ページの図 15 で図示するようなデータベース・メカニズムである場合には、データベース・トリガーは、コネクターの展開時に作成するアーカイブ表に、削除されたイベントをコピーすることができます。

保証付きイベント・デリバリー

保証付きイベント・デリバリーは、発生する可能性のあるいかなるサービス中断が発生した場合でも、金融取引などの重要なイベントが正しく処理されるようにします。この機能により、コネクター・フレームワークは、各イベントの検出と、ソース・コネクターのイベント・ストアから宛先コネクターの要求キューへの送信が、確実に 1 度のみ行なわれるようにすることができます。

保証付きイベント・デリバリーが実行されない場合には、コネクターがイベントを検出した時点から、必要なすべての処理が完了する時点までの間、障害が発生する可能性のある短い期間が生じます。この期間内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・レコードはイベント・ストアに残ったままになります。コネクターが再始動すると、コネクターはイベント・ストアでそのイベント・レコードを検出し、これを新しいイベントとして扱うため、このイベントは 2 回送信されてしまいます。

保証付きイベント・デリバリー、および配置するアダプターでこの機能を使用可能にする方法についての詳細は、アダプターのユーザー・ガイドを参照してください。

要求処理

注: WebSphere Business Integration Adapters の資料では、「要求処理」とは、統合ブローカーからコネクターに送信されたビジネス・オブジェクト要求メッセージの処理のことを指します。ハブ要求対話パターンとは、コネクターが結果を統合ブローカーに返すことを要求される要求処理のことです。ハブ片方向対話パターンとは、コネクターが結果を統合ブローカーに返すことを要求されない要求処理のことです。

統合ブローカーは、プロセスから要求を送信するよう通知されると、WebSphere MQ メッセージの形式でコネクターに要求を送信します。

通常のシナリオでは、統合ブローカーは、最初に、イベント通知ビジネス・オブジェクト・メッセージをソース・アプリケーションのコネクターから受信します。次に、このイベントを処理した結果として、要求ビジネス・オブジェクト・メッセージを宛先アプリケーションのコネクターに送信します。

宛先アプリケーションに対しては、次のいずれかを行なうことを要求できます。

- ビジネス・データを検索し、それを統合ブローカーに戻すこと。
- アプリケーションのデータ・ストアを更新すること。

例えば、統合ブローカーは、契約を削除する、パーツを更新する、顧客を作成するなどの要求メッセージをコネクターに送信することがあります。

コネクター・フレームワークは、統合ブローカーの要求を受け取ると、メッセージを適切なビジネス・オブジェクトに変換し、それをアプリケーション固有のコンポーネントに転送します。例えば、統合ブローカーが契約を削除する要求を送信する場合、アプリケーション固有のコンポーネントはその要求を Customer.Delete ビジネス・オブジェクトの形式で受け取ります。アプリケーション固有のコンポーネントは、そのビジネス・オブジェクトをアプリケーション要求 (通常、一組の API 呼び出し) に変換し、続いて、必要に応じて結果を戻します。

34 ページの図 16 に、ハブ要求対話パターンにおける、統合ブローカーからのメッセージの処理に関するコネクターの対話を示します。

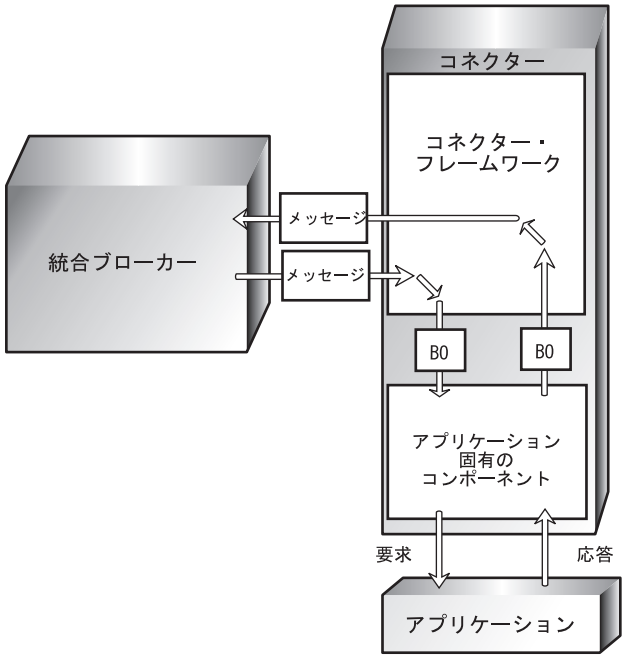


図 16. 要求の処理のためのコネクターの対話

アプリケーション固有のコンポーネントは、要求を受け取ると、次の 3 つのタイプの情報に基づいて要求の処理方法を決定します。

- ビジネス・オブジェクトの動詞
- ビジネス・オブジェクト定義自体に含まれ、ビジネス・オブジェクトの構成またはデコンストラクションで使用されるメタデータ
- 動詞に関するアプリケーション固有情報

以降の各トピックで、上記の要因について説明します。

動詞ベースの処理

コネクターのアプリケーション固有のコンポーネントは、そのアプリケーションのロジックおよび API に従って、要求に含まれている Create、Retrieve、Update、または Delete 動詞に対して応答します。さまざまなコネクターのアプリケーション固有のコンポーネントは、同じタイプの要求をそれぞれに異なる方法で処理しますが、結果は論理的に同じです。

一部のコネクターでは、要求に含まれる動詞のタイプとは無関係に、ビジネス・オブジェクトでの操作の実行に必要とするメソッドは 1 つのみです。ただし、多くのコネクターでは、それぞれの動詞ごとに、異なるメソッドが必要です。

アプリケーション固有のコンポーネントは、要求を受け取ると、そのアプリケーションのメソッドでビジネス・オブジェクトのアクティブな動詞に合致するものを起動します。例えば、アプリケーション固有のコンポーネントは、AppAEmployee.Update ビジネス・オブジェクトを受け取ると、AppAEmployee オブジェクト上で Update メソッドを起動します。Update メソッドは、更新を行なうためにアプリケーションと対話します。

図 17 に、動詞の処理メソッドをいくつか示します。

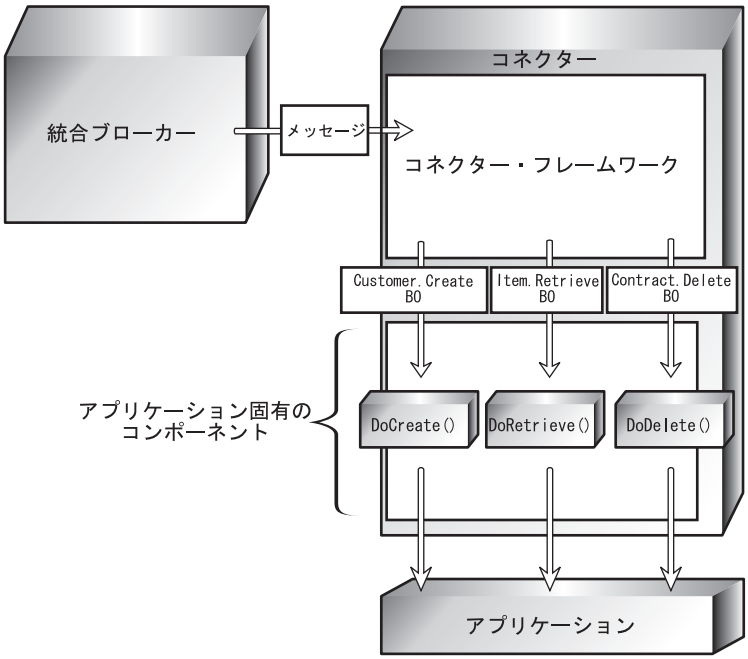


図 17. 要求の処理

図 17 で、コネクターは、Customer.Create、Item.Retrieve、または Contract.Delete 要求を受け取ると、それぞれに応じた DoCreate()、DoRetrieve()、または DoDelete() メソッドを起動します。各メソッドは、指定の操作を実行するために、そのアプリケーションに対応した適切なコマンドを作成します。

ビジネス・オブジェクトの構成とデコンストラクション

コネクターのアプリケーション固有のコンポーネントは、次に示すように、ビジネス・オブジェクトを構成およびデコンストラクションすることにより、そのイベント通知タスクおよび要求処理タスクを実行します。

- アプリケーション固有のコンポーネントは、それが統合ブローカーに送信しなければならないイベントを検出すると、そのイベントを表すビジネス・オブジェクトを構成します。
- アプリケーション固有のコンポーネントは、要求を表すビジネス・オブジェクトを統合ブローカーから受け取ると、そのビジネス・オブジェクトをデコンストラクションして、アプリケーション要求を作成します。

ビジネス・オブジェクト・メタデータおよびコネクターの動作

コネクターによる、アプリケーション・イベントからビジネス・オブジェクトへの変換およびビジネス・オブジェクトからアプリケーション要求への変換は、ビジネス・オブジェクトの設計時に定義されるデータ定義 (**メタデータ**) によって駆動されます。

アプリケーション固有のコンポーネントとビジネス・オブジェクト・メタデータは、組み合わせて使用するように設計されます。アプリケーション固有のコンポーネントとそのビジネス・オブジェクトの設計は、特定の機能がソフトウェアまたは

ハードウェアのいずれかによってインプリメントできる、コンピューター装置の設計に類似しています。開発者は、パフォーマンス、拡張性、および他の問題を考慮して、重要な機能の実装場所を決定します。

ビジネス・オブジェクト定義には、属性のタイプ、サイズ、およびデフォルト値を指定するプロパティーが組み込まれます。また、ビジネス・オブジェクト定義には、ビジネス・オブジェクトの処理方法に関しての、アプリケーション固有のコンポーネントに対する指示が含まれるアプリケーション固有のプロパティーも組み込まれます。

表 5. 属性に関連したアプリケーション固有情報の名前と値のペアのパラメーター例

パラメーター	説明
TN=*TableName*	データベース表の名前です。
CN=*col_name*	この属性のデータベース列の名前です。
FK=[..]fk_attributeName]	外部キー・プロパティーの値により親子関係が定義されます。
UID=AUTO	このパラメーターは、ビジネス・オブジェクト用の固有 ID を生成するように、またこの属性に値をロードするようにコネクターに通知します。
CA=*set_attr_name*	Copy Attribute (属性のコピー) プロパティーは、ある属性の値を別の属性にコピーするようコネクターに指示します。*set_attr*_name が現在の個々のビジネス・オブジェクト内の別の属性の名前に設定されている場合には、コネクターは、Create 操作でビジネス・オブジェクトをデータベースに追加する前に、指定された属性の値を使用して、この属性の値を設定します。

アプリケーション固有のコンポーネントは、ビジネス・オブジェクトを処理するとき、定義を読み取り、アプリケーション固有情報を使用してアプリケーション要求を作成します。ビジネス・オブジェクトの詳細については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

アプリケーション固有のコンポーネントはメタデータ主導型です。したがって、アプリケーション固有のコンポーネントのアクションは、ビジネス・オブジェクト定義にあるアプリケーション固有情報および他のメタデータによって制御されます。アプリケーション固有のコンポーネントには、サポートするビジネス・オブジェクトのタイプごとのハードコーディング命令はありません。アプリケーション固有のコンポーネントはメタデータ主導型であるため柔軟で、対応するアプリケーション・データがコネクターのメタデータ構文によって正確に記述できるかぎり、すべての新しいビジネス・オブジェクト定義、または変更したビジネス・オブジェクト定義すべてを自動的にサポートします。

ビジネス・オブジェクト作成の例

アプリケーション固有のコンポーネントがビジネス・オブジェクトをその定義から作成する手順を次に説明します。

1. ビジネス・オブジェクト定義をそのローカル・リポジトリーから取得し、それを使用してビジネス・オブジェクト・インスタンスを作成します。

2. アプリケーション固有のコンポーネントは、属性ごとにビジネス・オブジェクト・インスタンスをループし、アプリケーション固有情報を使用して、アプリケーション・エンティティーを取得するための API 呼び出しの作成、または照会のビルドを行います。
3. アプリケーション固有のコンポーネントは、要求をアプリケーションに送信し、結果を検索します。
4. アプリケーション固有のコンポーネントは、結果をループし、AppSpecificInfo の値を使用して、それぞれのビジネス・オブジェクト属性を表す検索値を判別します。

図 18 は、定義からビジネス・オブジェクトを作成するアプリケーション固有のコンポーネントの一例です。アプリケーション固有のコンポーネントは、キー値 (項目番号) が 123 である項目を含むアプリケーション・イベントを検索しています。アプリケーション固有のコンポーネントは、ビジネス・オブジェクト定義から Item ビジネス・オブジェクトを作成する必要があります。この定義には、Group、Description、Price、および ItemNum という 4 つの属性が含まれます。

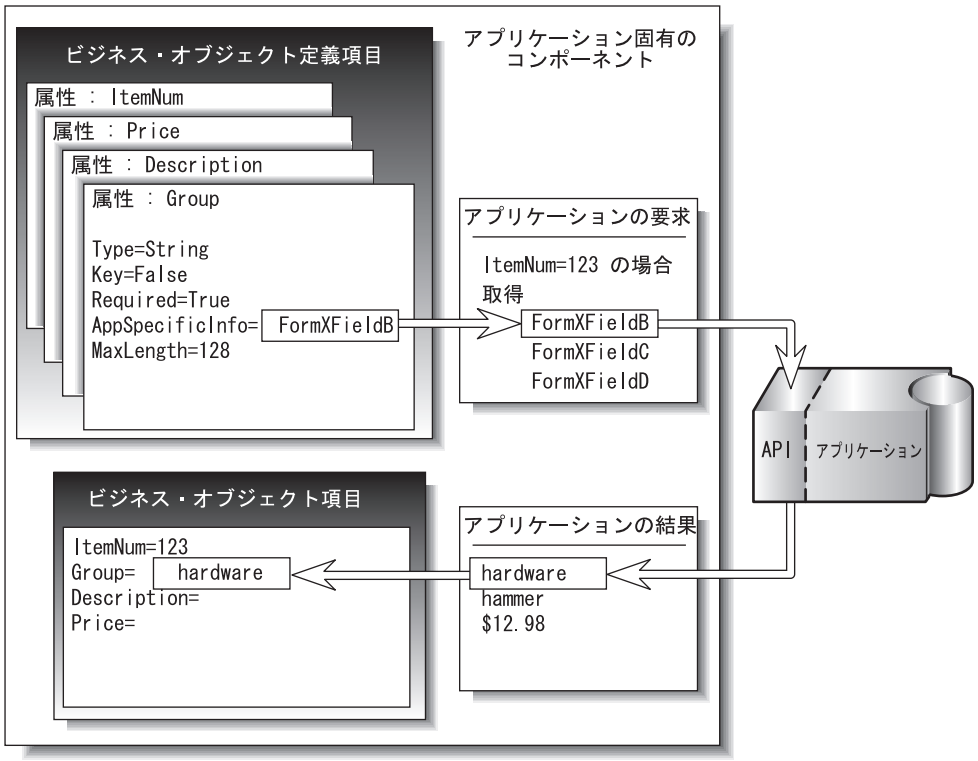


図 18. コネクター内でのビジネス・オブジェクトの作成

項目を識別するための項目番号 123 を使用して、アプリケーション固有のコンポーネントは残りの属性の値を検索します。アプリケーション固有情報は、必須データの用紙とフィールド ID を識別します。

例えば、FormXFieldB は Group データを識別します。アプリケーション固有のコンポーネントは、項目 ID 123 に関して用紙 X のフィールド B の値を要求します。続いて、アプリケーション固有のコンポーネントは戻り値「hardware」使用して、ビジネス・オブジェクトの Group 属性の値を入力します。

デコンストラクションのプロセスは、上記と正反対です。アプリケーション固有のコンポーネントは、ビジネス・オブジェクト定義を使用して、コンポーネントが受け取ったビジネス・オブジェクトに含まれるデータに基づいてアプリケーション要求を作成する方法を決定します。

動詞に関するアプリケーション固有情報

ビジネス・オブジェクト定義内のそれぞれの動詞には、それに関連付けられた、アプリケーション固有情報を持たせることができます。動詞の持つアプリケーション固有情報の内容は、その特定のコネクターに固有です。動詞に関するアプリケーション固有情報は、コネクターのアプリケーション固有のコンポーネントに、ビジネス・オブジェクトを処理するための追加の指示を与えます。

例えば、ビジネス・オブジェクト定義内の Retrieve 動詞に関するアプリケーション固有情報では、そのアプリケーション固有のコンポーネント内の Retrieve メソッドに特別な入力引き数が与えられる場合があります。

一例として、MyApp アプリケーションで InventoryItem に関する情報が次の 3 つの用紙で表示される場合を想定します。

- InventoryItem-New
- InventoryItem-Change
- InventoryItem-Remove

MyApp 用のアプリケーション固有のコンポーネントは、インベントリー項目で操作を行なうとき、その操作に対応する正しいフォームを参照する必要があります。InventoryItem ビジネス・オブジェクト定義では、動詞に関するアプリケーション固有情報を使用してフォーム名を格納することができます。

動詞固有のメソッドと、それらメソッドへのアプリケーション固有の入力とを組み合わせると、アプリケーション固有のコンポーネントに固有の処理の指示を与えることができます。

コネクター構成

コネクターを使用するにはその前に、Connector Configurator を使用して、次のものを含む構成ファイルを定義する必要があります。

- コネクターがサポートするビジネス・オブジェクト
- コネクターの構成プロパティー

コネクター構成プロパティーには、**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**という、2 つのタイプがあります。

標準プロパティーは、すべてのコネクターに適用され、次に示すような情報を指定します。

- コネクターで使用する統合ブローカー
- コネクターのローカル・リポジトリーのロケーション
- コネクターによって使用されるキューを管理するキュー・マネージャーの名前

アプリケーション固有のプロパティーでは、特定のアプリケーション固有のコンポーネントがアプリケーションとのセッションを確立するために必要とする値が指定されます。また、アプリケーション固有のコンポーネントの処理の振る舞いの一部について、指図します。次に、さまざまなコネクターでのアプリケーション固有のプロパティーの例を示します。

- アプリケーションを実行するマシンの名前または IP アドレス
- アプリケーション・データベースの名前
- アプリケーションへのアクセスでコネクターが使用する必要がある、ログイン・ユーザー ID とパスワード
- イベント受信箱の名前
- ポーリング・イベントごとの、検索するイベントの数

一部のコネクター構成プロパティーは、コネクターの開始時にコマンド行で設定することもできます。 コマンド行で設定するプロパティーは、コネクターの構成ファイルで設定された値をオーバーライドします。

コネクターの構成の詳細については、 64 ページの『コネクターの構成』を参照してください。コネクターの開始の詳細については、 117 ページの『コネクターの始動』を参照してください。

コネクターの開発

コネクターを変更または作成するには、クラス・ライブラリーおよびサンプルを使用して、アプリケーション固有のコンポーネントを作成します。続いて、Connector Configurator を使用して、ビジネス・オブジェクト定義を作成および変更し、コネクター・リポジトリー定義を作成します。

コネクター開発には、アプリケーション固有のコンポーネントと特定のアプリケーションとの関係の定義が含まれます。アプリケーション固有のコンポーネントの機械語コーディングは、通常、比較的簡単な処理です。最も難しいのは次のタスクです。

- アプリケーションのイベント通知メソッドの設計
- ビジネス・オブジェクト定義の定義
- ビジネス・オブジェクトとアプリケーション・オブジェクトとの関係の定義

コネクターのアーキテクチャー、変更、および開発の詳細については、「*コネクター開発ガイド (Java 用)*」または「*コネクター開発ガイド (C++ 用)*」を参照してください。

次の章の 41 ページの『第 4 章 データ移送および統合ブローカー』では、ビジネス・インテグレーション・システムのコンポーネント間での情報の伝達に使用する、メカニズムとプロトコルについて詳細に説明します。

第 4 章 データ移送および統合ブローカー

WebSphere Application Server が統合ブローカーとなるように実装されている WebSphere Business Integration Adapters では、コネクター・フレームワークと統合ブローカーの間の通信移送機構として、WebSphere MQ キューおよび Java Messaging Service (JMS) ソフトウェアを使用します。WebSphere MQ Queue マネージャーが管理する定義済みキューに対して、コネクターがメッセージの読み書きを行います。

本章では、WebSphere Business Integration システムのアプリケーション間データ交換に用いられるメッセージング・インターフェースおよびプロトコルについて詳細に説明します。

統合ブローカーの役割

統合ブローカーは、アプリケーション・コネクター間のメッセージの経路指定とメッセージ処理の両方を実行します。具体的には、以下の処理を実行します。

- コネクターのアウトバウンド・キューからメッセージを受信する。
- メッセージの処理を実行する。
- この処理の結果、1 つ以上のコネクターのインバウンド・キューにメッセージを送信する。

統合ブローカーとコネクター間の通信は、**非同期**または**同期**の、いずれかになります。

非同期データ移送

非同期メッセージング移送を使用するプログラムでは、接続の確立やメッセージ (個別単位ごとのデータ) 待ちを必要としません。各プログラムは、メッセージング・サービスと非同期で対話することによってメッセージの送受信を行います。メッセージング・サービスは、宛先プログラムが使用不可の場合にメッセージを保管したり、そのプログラムが使用可能になるまで再試行したりすることによって、メッセージの送達が確実にできるようにします。

同期データ移送

同期メッセージング移送を使用するプログラムでは、統合ブローカーへの要求メッセージを同期要求キューに配置し、統合ブローカーからの応答メッセージを同期応答キューで受け取ります。応答メッセージの相関 ID によって、どの要求メッセージに応答しているかが識別されます。一般に応答メッセージは、ビジネス・オブジェクト・メッセージと、要求メッセージが正常に処理されたかどうかを示す状況表示で構成されています。

メッセージ交換のインターフェース

このセクションでは、メッセージ自体と、メッセージの処理に必要な情報とを送信するためにコネクター・フレームワークおよび統合ブローカーで使用されるメッセージング・インターフェースについて説明します。

ビジネス・インテグレーション・システムにおいては、何種類ものメッセージが交換されています。メッセージの異なるタイプを識別するため、またメッセージを正しく処理して送信するために、必要不可欠な情報が、各メッセージのメッセージ・ヘッダーおよびメッセージ・ディスクリプターに格納されています。ビジネス・インテグレーション・システム向けに作成したメッセージ・フローは、処理要求のあったメッセージを認識し、正しく管理するために、下記の情報を使用します。

以下のタイプのメッセージが渡されます。

- イベント・デリバリー・メッセージは、送信元アプリケーションで発生したイベントの通知として、コネクター・フレームワークから統合ブローカーに送信されます。WebSphere Application Server 環境では、通常、このようなメッセージはエージェント・デリバリー・メッセージと呼ばれています。エージェント・デリバリー・メッセージは、コネクター・フレームワークによって DeliveryQueue (後の『必要なキューのタイプ』を参照) に書き込まれた後、統合ブローカーによって DeliveryQueue から読み取られます。
- 要求メッセージは、データ要求を伝達するためにコネクター・フレームワークと統合ブローカーの間で交換されます。

  統合ブローカーがコネクター・フレームワークに要求を送信して応答を待機する場合、このメッセージ交換はハブ要求と呼ばれます。ハブ要求では、統合ブローカーからの要求は RequestQueue に書き込まれ、そこからコネクター・フレームワークによって読み取られます。

  コネクター・フレームワークから統合ブローカーに送信される要求は、エージェント要求と呼ばれます。エージェント要求は、コネクターによって SynchronousRequestQueue に書き込まれ、そこから統合ブローカーによって読み取られます。

  統合ブローカーからの要求でも、コネクター・フレームワークからの要求でも、要求のメッセージ・ヘッダーに含まれる JMSReplyto プロパティーにより、応答の要否が判別され、応答の送信先キューが指定されます。
- 応答メッセージは、データ要求に対する応答として、コネクター・フレームワークと統合ブローカーの間で交換されます。コネクター・フレームワークが統合ブローカーからの要求に応答するときには、ResponseQueue に応答メッセージが書き込まれます。統合ブローカーがコネクター・フレームワークからの要求に応答するときには、SynchronousResponseQueue に応答メッセージが書き込まれます。
- 片方向メッセージは、統合ブローカーからコネクター・フレームワークに送信されます。WebSphere Application Server 環境では、通常、このようなメッセージはハブ片方向メッセージと呼ばれています。ハブ片方向メッセージは、統合ブローカーによって RequestQueue に書き込まれ、コネクター・フレームワークによって RequestQueue から読み取られます。
- 管理メッセージは、管理コマンドを伝達するためにコネクター・フレームワークと統合ブローカーの間で交換されます。

メッセージ・フォーマット

コネクター・フレームワークと統合ブローカーの間で交換されるメッセージは、データ・ハンドラーが以下の項目に基づいてフォーマット設定します。

- コネクターの構成ファイル内の WireFormat 標準プロパティー
- メッセージ本文のフォーマットを詳述する XML スキーマ
- メッセージの内容: ビジネス・オブジェクトまたは管理メッセージ
- メッセージの発信元および宛先

各メッセージには、メッセージ・ディスクリプター (MQMD)、メッセージ・ヘッダー (MQRFH2)、およびメッセージ本文という 3 つのコンポーネントが含まれています。

メッセージ・ディスクリプター

WebSphere MQ メッセージ・ディスクリプター (MQMD) には、メッセージ ID およびメッセージの処理に必要な情報が入っています。

メッセージ・ヘッダー

MQRFH2 メッセージ・ヘッダーは、メッセージ内容に関連した JMS 固有のデータを搬送します。また、直接 JMS と関連付けのない追加情報も搬送できます。

メッセージ本文

メッセージ本文は、そのメッセージ用に指定された XML スキーマの内容に従ってフォーマット設定されます。メッセージをフォーマット設定するために正しい XML スキーマをデータ・ハンドラーが検出および使用するには、次の 3 つの名前が一致しなければなりません。

- コネクターのリポジトリーに格納されている XML スキーマの名前。
- 統合ブローカーのメッセージ・リポジトリーにインポートされ、メッセージ・セット定義として保管されている XML スキーマの名前。
- メッセージの MQRFH2 メッセージ・ヘッダーにある messagetype の内容。

メッセージ・フォーマットおよび特定のプロパティーの設定値を 139 ページの『付録 A. WebSphere MQ のメッセージ・フォーマット』に示します。これらは、コネクター・フレームワークおよび統合ブローカーによって交換される各種のメッセージに使用されます。

メッセージ・キュー

ここでは、コネクターと共に使用するための定義および構成が必要な WebSphere MQ キューについて説明します。

必要なキューのタイプ

コネクター・フレームワークと WebSphere Application Server 間でのビジネス・オブジェクト・メッセージおよび管理メッセージの移送には、各 WebSphere MQ メッセージ・キューのセットを個別に使用します。これらのキューの詳細については、60 ページの『WebSphere MQ キューの作成』を参照してください。

キュー・マネージャー

コネクターは、1 つのキュー・マネージャーを使用して、キューとのすべての対話を管理しています。コネクターの構成ファイル内の標準プロパティーには、始動時にコネクターが必要とするキュー・マネージャー情報が入っています。コネクターはこの情報を使用して、統合ブローカーとの通信に用いるキュー・マネージャーとの接続を確立します。

WebSphere Business Integration システムでは、いくつかのキュー・マネージャーとキュー構成をサポートしています。コネクターは、以下のどのモードにおいても、キュー・マネージャーと通信することができます。

- **バインディング・モード:** 統合ブローカーおよびコネクターは、TCP/IP 接続を使用せずに、直接キュー・マネージャーとの通信を行います。キュー・マネージャーおよびコネクターは、同一のマシン上に存在し、同一のキュー・マネージャーを共用する必要があります。デフォルトでは、このバインディング・モードになります。
- **リモート・キュー定義付きバインディング・モード:** 統合ブローカーとコネクターが異なるマシンにインストールされ、各マシンで専用のキュー・マネージャーが稼働している場合でも、コネクターおよび統合ブローカーはバインディング・モードでそれぞれのキュー・マネージャーと通信できますが、それにはリモート・キュー定義が必要です。
- **クライアント・モード:** TCP/IP を基本トランスポートとして使用するクライアント接続を介して通信が行われます。キュー・マネージャーとコネクターが異なるマシン上にある場合、コネクターが使用できるのはクライアント・モードのみです。

詳細

WebSphere MQ メッセージの詳細については、「*WebSphere MQ: Java の使用*」を参照してください。WebSphere MQ キューの詳細については、「*WebSphere MQ: 相互通信*」および「*WebSphere MQ: MQSC リファレンス*」を参照してください。

第 2 部 展開と管理

第 5 章 概要: 統合ブローカーとしての WebSphere Application Server の実装

WebSphere Application Server は、統合ブローカーの役割を果たすときには、J2EE テクノロジーを使用して、ネットワーク経由またはインターネット経由で異種アプリケーション間データ交換を実現します。

WebSphere Application Server 環境では、ビジネス・データの交換のために、4 つのタイプの操作がサポートされています。これらの操作のタイプは、「対話パターン」と呼ばれることがあります。次の表では、対話パターンと、対話パターンを実装するために使用する WebSphere MQ キューをそれぞれ説明します。

対話パターン	説明	使用するキュー
エージェント要求	コネクターは、同期要求メッセージを WebSphere Application Server に送信して、アプリケーションにイベントが発生したことをレポートし、応答を待ちます。(個々のアダプターの資料では、このことを通常「*同期イベント・デリバリー*」と呼びます。)	ブローカーに対する要求の場合は SynchronousRequestQueue。 コネクターに対する要求の場合は SynchronousResponseQueue。
エージェント・デリバリー	コネクターは、非同期メッセージをブローカーに送信して、アプリケーションにイベントが発生したことをレポートします。コネクターは応答を待ちません。(個々のアダプターの資料では、このことを通常「*非同期イベント・デリバリー*」と呼びます。)	DeliveryQueue
ハブ要求	ブローカーは同期要求メッセージをコネクターに送信して、応答を待ちます。(個々のアダプターの資料では、このことを通常「*同期要求処理*」と呼びます。)	ブローカーからコネクターへのメッセージの場合は RequestQueue。 コネクターからブローカーへのメッセージの場合は ResponseQueue。
ハブ片方向	ブローカーは非同期要求メッセージをコネクターに送信しますが、応答は待ちません。(個々のアダプターの資料では、このことを通常「*非同期要求処理*」と呼びます。)	RequestQueue

これらのメッセージ交換を行う統合ブローカーとして WebSphere Application Server を実装するために必要な作業は、概説すると次のようになります。

- 統合コンポーネントの設計と開発

  統合コンポーネントとは、アプリケーションと相互作用する WebSphere Business Integration Adapter コネクターや、交換対象データを格納する WebSphere Business Integration Adapter ビジネス・オブジェクトのことです。本書の第 1 部では、統合コンポーネントについて理解するために必要な概念を紹介し、WebSphere Business Integration Adapter のコネクターとビジネス・オブジェクトの設計と開発についてさらに詳しく解説した資料へのリンクを示しています。
- 統合コンポーネントのインストールと構成

  コネクターをインストールするために必要な情報は、後の 49 ページの『第 6 章 WebSphere Business Integration Adapters のインストール』 の章と、この章で紹介するその他の資料で提供されています。コネクターおよびそのコネクターのビジネス・オブジェクトを構成するための情報 (キューの設定作業や Connector Configurator を使用した作業など) は、本書で後述します。
- WebSphere Application Server への構成済みコンポーネントの配置

  この作業には、System Manager を使用して構成済みコンポーネントからユーザー・プロジェクトを作成する作業や、WebSphere Application Server に配置できるサービス・プロジェクトを WebSphere Studio Application Developer を使用して作成する作業が含まれます。これらの作業については、本書で後述します。

第 6 章 WebSphere Business Integration Adapters のインストール

本章では、WebSphere Business Integration Adapters のインストール方法と、WebSphere Application Server を統合ブローカーとして使用するサポート・ソフトウェアについて説明します。

以前のバージョンから WebSphere Business Integration Adapters をアップグレードする場合の手順については、219 ページの『付録 G. WebSphere Business Integration Adapters のアップグレード』を参照してください。

本章では、Windows オペレーティング・システムおよび UNIX オペレーティング・システムでのインストール方法を説明します。以下のうち、ご使用のオペレーティング・システムに該当するセクションを参照してください。

- 『Windows システムへのインストール』
- 51 ページの『UNIX システムへのインストール』

本章には、統合ブローカーにサポート・ソフトウェア (JDK および WebSphere MQ) をインストールする場合に固有の情報が含まれています。別の資料「*WebSphere Business Integration Adapters インストール・ガイド*」には、WebSphere Business Integration Adapters のインストールに必要なすべての詳細情報が記載されています。Windows または UNIX システムへのインストールについて説明している本章のセクションでは、インストール処理の適切な箇所で「*WebSphere Business Integration Adapters インストール・ガイド*」を参照してください。

Windows システムへのインストール

このセクションのトピックは以下のとおりです。

- 『ソフトウェア要件』
- 50 ページの『JDK のインストール』
- 50 ページの『WebSphere MQ のインストール』
- 51 ページの『WebSphere Business Integration Adapters のインストール』
- 51 ページの『WebSphere Studio Application Developer Integration Edition へのプラグインのインストール』

ソフトウェア要件

WebSphere Business Integration システムは、WebSphere Business Integration Adapters およびその他のコンポーネントから構成されます。WebSphere Business Integration Adapters および一部の関連コンポーネントは、CD または ESD (IBM Electronic Software Delivery) で配布されます。WebSphere Business Integration システムで使用されるその他のコンポーネントは、個別に入手し、インストールする必要があります。

表 6 は、WebSphere Business Integration システムのソフトウェア要件を一覧表示したものです。

表 6. WebSphere Business Integration システム・ソフトウェア要件 (Windows システムの場合)

ソフトウェア	製品に同梱
Windows 2000 (Professional、Server、または Advanced Server) Service Pack 4 **注:** WebSphere Business Integration Adapter Framework 2.4.0 のリリースより、アダプターは Windows NT ではサポートされません。	なし
IBM WebSphere Application Server Enterprise Edition バージョン 5.0.2	なし
WebSphere Studio Application Developer Integration Edition バージョン 5.0.1	なし
Java Runtime Environment (JRE): IBM バージョン 1.3.1 SR5。	あり
Java Development Kit (カスタム Java コネクターを開発する場合にのみ必要)。IBM JDK バージョン 1.3.1 SR5。	あり
IBM WebSphere MQ バージョン 5.3.0.5 (CSD05 が適用されたバージョン 5.3)	なし
ブラウザー: HTML 文書を参照するために、Microsoft Internet Explorer や Netscape Navigator などの HTML ブラウザーが必要です。サポートされている正確なバージョンについては、`http://www.ibm.com/integration/wbiadapters/library/infocenter` からダウンロード可能な説明を参照してください。	なし

JDK のインストール

JDK (Java Development Kit) は、カスタム Java コネクターを開発する (Java コンパイラーが必要) 場合のみ必要です。Windows システムでの開発には、JDK は WebSphere Business Integration Adapter Framework に含まれています。JDK のインストール情報については、「*WebSphere Business Integration Adapters インストール・ガイド*」を参照してください。

WebSphere MQ のインストール

IBM WebSphere MQ は、統合ブローカーとアダプターの間の通信を可能にするメッセージング・ソフトウェアです。

インストールおよび構成情報については、次の WebSphere MQ 各資料を参照してください。

- *WebSphere MQ: スタートアップ・ガイド*
- *WebSphere MQ: システム管理*
- *WebSphere MQ: 相互通信*

注: これらの資料は次の IBM Web サイトよりブラウズまたはダウンロードできます。www.ibm.com/software/integration/mqfamily

WebSphere MQ を統合ブローカーと連動するように構成する方法については、60 ページの『WebSphere Application Server 用の WebSphere MQ キューの構成』を参照してください。

WebSphere Business Integration Adapters のインストール

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

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

WebSphere Studio Application Developer Integration Edition へのプラグインのインストール

WebSphere Application Server を、WebSphere Business Integration Adapters と組み合わせて統合ブローカーとして使用するには、特定のプラグインを WebSphere Studio Application Developer Integration Edition の既存のインストール・システムに追加する必要があります。このプラグインは、WebSphere Business Integration Adapter Framework のインストーラーを実行すると、システムにコピーされます (51 ページの『WebSphere Business Integration Adapters のインストール』を参照)。Adapter Framework のインストーラーは、次のプラグインを指定した場所にコピーします。

¥plugins¥com.ibm.btools.adaptermonitor
¥plugins¥com.ibm.btools.adaptermonitor.help
¥plugins¥com.ibm.btools.csm
¥plugins¥com.ibm.btools.csm.help
¥plugins¥com.ibm.btools.itools.ae.settings
¥plugins¥com.ibm.btools.itools.codeGen
¥plugins¥com.ibm.btools.itools.cwconverter
¥plugins¥com.ibm.btools.itools.datamanager
¥plugins¥com.ibm.btools.itools.msg.importer
¥plugins¥com.ibm.btools.itools.wsdlgen

インストーラーがプラグイン・ファイルをシステムの正しい場所にコピーするためには、Adapter Framework インストーラーを実行する前に WebSphere Studio Application Developer Integration Editions をインストールしておく必要があります。WebSphere Studio Application Developer Integration Edition のインストールについての一般的な説明については、Infocenter に用意されているこの製品に関する資料を参照してください。

UNIX システムへのインストール

このセクションのトピックは以下のとおりです。

ソフトウェア要件

WebSphere Business Integration システムは、WebSphere Business Integration Adapters およびその他のコンポーネントから構成されます。WebSphere Business Integration Adapters および一部の関連コンポーネントは、CD または ESD (IBM Electronic Software Delivery) で配布されます。WebSphere Business Integration システムで使用されるその他のコンポーネントは、個別に入手し、インストールする必要があります。

表 7 は、WebSphere Business Integration システムのソフトウェア要件を一覧表示したものです。

表 7. WebSphere Business Integration システム・ソフトウェア要件 (UNIX システムの場合)

ソフトウェア	製品に同梱
オペレーティング・システム: • **AIX** - AIX 5L バージョン 5.1 (保守レベル 1)、または バージョン 5.2 (保守レベル 4) • **Solaris** - Solaris 7.0 または 8.0 (Patch Cluster 2003/7/23) • **HP-UX**- HP-UX 11.11 (11i,r=B.11.110306.4)。 2003 年 6 月 GOLDBASE 11i,r=B.11.110306.4 および 2003 年 6 月 GOLDAPPS11i 11i,r=B.11.110306.4 のバンドル	なし
IBM WebSphere Application Server Enterprise Edition バージョン 5.0.2	なし
Java ランタイム環境: • **AIX** - IBM JRE バージョン 1.3.1 SR5。 • **Solaris** - Sun JRE バージョン 1.3.1.7。 • **HP-UX** - Sun JRE バージョン 1.3.1.6。	あり
Java Development Kit (カスタム Java コネクターを開発する場合にのみ必要)。 • **AIX:** IBM JDK バージョン 1.3.1 SR5。 • **Solaris:** Sun JDK バージョン 1.3.1.7。 • **HP-UX:** Sun JDK バージョン 1.3.1.6。	なし
WebSphere MQ バージョン 5.3.0.5 (CSD05 が適用されたバージョン 5.3)	なし
ブラウザー: HTML 文書を参照するために、Microsoft Internet Explorer や Netscape Navigator などの HTML ブラウザーが必要です。サポートされている正確なバージョンについては、`http://www.ibm.com/integration/wbiadapters/library/infocenter` からダウンロード可能な説明を参照してください。	なし

JDK のインストール

JRE には、WebSphere Business Integration Adapters を実行するために必要な Java 仮想マシン (JVM) が組み込まれています。ただし、JavaC (Java コンパイラー) などの開発ツールは組み込まれていません。カスタム・コネクターを作成しない 場合は、JRE を使用する。

JRE には、WebSphere Business Integration システムの実行に必要な Java ソフトウェアのランタイム・コンポーネントが含まれています。JRE は WebSphere Business Integration Adapter Framework に含まれています。したがって、JRE を個別にインストールする必要はありません。

カスタム・コネクターを作成する場合は、完全な JDK をインストールしてください。これは、Sun または IBM の Web サイトからダウンロードできます。Solaris

および HP-UX の場合、JDK は http://java.sun.com/products/jdk/1.3 からダウンロードできます。AIX の場合、JDK は http://www.ibm.com/developerworks/java/jdk からダウンロードできます。

JDK を、現在のシステムにインストールするには、次の手順に従います。

1. JDK のインストール先となるディレクトリーへ移動する。

   ```
   cd /install_dir
   ```

   ここで、*install_dir* は JDK ソフトウェアがインストールされるディレクトリーのパスです。このソフトウェアは任意の場所にインストールできます。

   - JDK は、通常 /usr ディレクトリーのサブディレクトリーにインストールされます。/usr ファイル・システムに十分なスペースがある場合は、ここに JDK をインストールします (例えば、/usr/jdk1.3)。
   - JDK の抽出ディレクトリー (例えば、/*home_dir*/jdk1.3) を作成して、このディレクトリーに移動することもできます。/usr/jdk1.3 ディレクトリーからこの抽出ディレクトリーへのシンボリック・リンクを作成する必要があります。
2. Sun または IBM の Web サイトで、JDK をダウンロードする Web ページを検索する。

   JDK は通常、Java 2 SDK Standard Edition for UNIX の下にあります。
3. 付属の README ファイルをダウンロードして、JDK のダウンロード方法を確認する。
4. JDK を、インストール用ディレクトリーにダウンロードする。

   IBM および Sun では、自己解凍機能のある tar ファイルを用意しており、その内容を任意のディレクトリーへ格納することができます。このファイルは、通常は *JDK_version*.bin のフォーマットになっています (ここで、*JDK_version* はダウンロードする JDK のバージョンです)。このファイルから JDK ソフトウェアを取り出す方法については、README ファイルを参照してください。
5. JDK についてリストされたオペレーティング・システム用パッチの中から、ご使用の UNIX オペレーティング・システムのバージョンに適したものをダウンロードする。

   これらのパッチをインストールする前に、まだパッチがインストールされてはいないことを確認してください。
6. ダウンロード・ディレクトリー (/usr/jdk1.3 ディレクトリーなど) から Java ディレクトリーへのリンクを作成する。

   a. rm /usr/java

   b. ln -s /usr/jdk1.3 /usr/java

7. インストール後、JDK の bin ディレクトリーが PATH 環境変数に組み込まれていることを確認する。

   これは、次のいずれかの方法で行います。

   - ユーザーの始動時に読み込まれる、/etc/profile を編集する。 ksh、bash、および sh などのシェルは、/etc/profile ファイル内の設定値を使用します。

     例えば、JDK を /usr/jdk1.3 ディレクトリーにインストールする場合は、/etc/profile の PATH エントリーを次のように編集します。

     ```
     PATH=/usr/jdk1.3/bin:$PATH
     export PATH
     ```

   - WebSphere Business Integration 管理者のホーム・ディレクトリーにある、個人用プロファイル・ファイルを編集する。

     この個人用プロファイル・ファイルの名前は、WebSphere Business Integration 管理者のアカウントで使用されている特定のシェルによって異なります。この個人用プロファイルに対する変更は、WebSphere Business Integration 管理者としてログインしたユーザーのみ に影響します。

     特定のシェルに最適な構文を使用して、PATH 環境変数に JDK bin ディレクトリーを組み込みます。例えば、WebSphere Business Integration 管理者のアカウントが sh シェルを使用している場合、JRE パス (/usr/java/bin) を追加する行を次のように編集して、JDK パスを組み込みます。

     ```
     PATH=/usr/jdk1.3/bin:/usr/java/bin:$PATH
     export PATH
     ```

     これらの行を、WebSphere Business Integration 管理者のアカウントの個人用プロファイル・ファイルで CWSharedEnv.sh ファイルを指定している行の後 に追加します。

WebSphere MQ のインストール

IBM WebSphere MQ は、統合ブローカーとアダプターの間の通信を可能にするメッセージング・ソフトウェアです。

インストールおよび構成情報については、次の WebSphere MQ 各資料を参照してください。

- *WebSphere MQ: スタートアップ・ガイド*
- *WebSphere MQ: システム管理*
- *WebSphere MQ: 相互通信*

注: これらの資料は次の IBM Web サイトよりブラウズまたはダウンロードできます。http://www.ibm.com/software/integration/mqfamily

WebSphere MQ を統合ブローカーと連動するように構成する方法については、60 ページの『WebSphere Application Server 用の WebSphere MQ キューの構成』を参照してください。

WebSphere Business Integration Adapters のインストール

WebSphere Business Integration Adapters をインストールするには、まずサポートされるバージョンの製品を取得します。次に、Web アドレス http://www.ibm.com/software/websphere/wbiadapters/infocenter の WebSphere Business Integration Adapters InfoCenter から、そのリリースの資料をダウンロードしてください。「*WebSphere Business Integration Adapters インストール・ガイド*」の手順に従って、ご使用のビジネス・インテグレーション・システムにインストールしてください。

第 7 章 WebSphere Business Integration システムの構成

本章では、WebSphere Application Server と共に使用するコネクターのセットアップに必要な作業について説明します。この説明は、以下のことを前提としています。

- WebSphere Application Server Enterprise v. 5.0.2 以降、および WebSphere Studio Application Developer Integration Edition v. 5.0.1 以降をインストール済みであること。
- WebSphere Business Integration Adapter および Adapter Framework、およびその他に必要なアダプター用ソフトウェア (例えば、あるアダプターには必要でその他のアダプターでは不要なデータ・ハンドラーなど) がインストール済みであること。
- 本書の『WebSphere Business Integration Adapters for Windows のインストール』の章の『ソフトウェア要件』に記載のとおりに、MQSeries およびその他の前提ソフトウェアをインストール済みであること。
- コネクターを作成または入手して、ご使用のシステムにインストール済みであること。コネクターのインストールに関する説明については、入手したコネクターに付属している資料を参照してください。独自のコネクターを作成した場合には、「コネクター開発ガイド」を参照してください。

本章では、WebSphere Application Server、ビジネス・オブジェクト、およびコネクターを、ビジネス・インテグレーション・システムとして連携するように構成する方法について説明します。本章は、次の各セクションから構成されています。

構成タスクの概要

ビジネス・インテグレーション・システムを構成するには、以下のタスクを実行する必要があります。

1. ビジネス・オブジェクト定義を作成し、作成した定義を System Manager 内の統合コンポーネント・ライブラリー (ICL) プロジェクトに追加する。
2. 実装しようとしているデータ・トランザクションに必要な JMS キューを定義して、コネクターがサポートされるように WebSphere Application Server を構成する。WebSphere Application Server の管理コンソールを使用してキューの JNDI 名を設定してから、Business Integration パースペクティブの配置記述子エディターを使用して名前のマッピングを行う。

3. コネクターと共に使用するビジネス・オブジェクト定義をまだ生成していない場合は、生成する。
4. System Manager 内でビジネス・オブジェクト定義を追加する。
5. Connector Configurator を使用して、関連するビジネス・オブジェクトをサポートするようにコネクターを構成する。
6. 標準およびアプリケーション固有のコネクター構成プロパティー (キューを指定するプロパティーを含む) を構成する。
7. コネクター用のトレース、ロギング、およびメッセージングの各オプションを構成する。

次に、これら各タスクの詳細について説明します。

上記のタスクの完了後に行う次のステップでは、System Manager と WSAD-IE を使用して、構成済みコンポーネントを基にサービス・プロジェクトを作成します。また、作成したサービス・プロジェクトを WebSphere Application Server に配置します。この作業については、75 ページの『第 8 章 WebSphere Application Server への配置』で説明します。

ビジネス・オブジェクト定義の作成

コネクターで使用されるビジネス・オブジェクト定義を作成する際には、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

注: WebSphere Application Server と共に使用される WebSphere Business Integration Adapters では、アプリケーション固有のビジネス・オブジェクトのみが使用され、汎用ビジネス・オブジェクトは使用されません。本書でビジネス・オブジェクトというときはすべて、アプリケーション固有のビジネス・オブジェクトを意味しています。汎用ビジネス・オブジェクトは、WebSphere InterChange Server 統合ブローカーを基にしたビジネス・インテグレーション・システムで使用されます。「*ビジネス・オブジェクト開発ガイド*」など、WebSphere Business Integration Adapters ライブラリーの一部の資料は、WebSphere InterChange Server ライブラリーの一部をなすものでもあるため、両タイプのビジネス・オブジェクトに言及しています。

ビジネス・オブジェクト定義を作成するためのオプションは以下のとおりです。

- ODA (Object Discovery Agent) を使用して、アプリケーション固有のビジネス・オブジェクト定義を生成する。ODA により、アプリケーション内の指定されたオブジェクトが検査され、ビジネス・オブジェクト属性に対応するこれらのオブジェクト属性の要素とその属性が「発見」され、情報を表現するビジネス・オブジェクト属性定義が生成されます。 Business Object Designer には、Object Discovery Agent にアクセスしたり、そのエージェントと対話式に作業を行ったりするためのグラフィカル・インターフェースが用意されています。構成するコネクターに ODA (Object Discovery Agent) が用意されているかどうかを判別するには、当該コネクター用のアダプター・ユーザー・ガイドを参照してください。
- ODA がビジネス・インテグレーション・アダプターに組み込まれていない場合は、ODK (Object Discovery Agent Development Kit) を使用して ODA を開発してから、それをアプリケーションに対して実行する。

- Business Object Designer ツールを使用して、ビジネス・オブジェクト定義を手動で作成する。

さらに、アダプターの多くにはサンプルのビジネス・オブジェクトが付属しています。付属している場合、サンプルは次の製品ディレクトリー内に存在しています。

ProductDir¥connectors¥*ConnName*¥Samples

Business Object Desginer を始動するには、「スタート」メニューの「ファイル名を指定して実行」を選択し、参照によって BusObjDesigner.exe ファイルを探し出して、実行します。この方法を使用する代わりに、BusObjDesigner.exe への Windows ショートカットを作成して使用してもかまいません。

Business Object Designer の使用についての詳細は、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

アプリケーション固有のビジネス・オブジェクト用のビジネス・オブジェクト定義を作成し終えたら、203 ページの『付録 F. Visual Test Connector の使用』をお読みください。これには、ビジネス・オブジェクト定義の作成後にそれらの定義をテストする方法が説明されています。

ICL へのビジネス・オブジェクトの追加

コネクターがビジネス・オブジェクトを利用できるようにするには、そのビジネス・オブジェクトを、そのコネクターでサポートされるビジネス・オブジェクトに指定する必要があります。サポートされるビジネス・オブジェクトの指定は、Connector Configurator を使用してコネクターを構成するときに行わなければなりません。これを行うには、まず、そのコネクターが含まれている統合コンポーネント・ライブラリーと同じ統合コンポーネント・ライブラリーに、ビジネス・オブジェクト定義ファイルを追加する必要があります。

この作業は、System Manager を使用して、以下の手順で行います。

1. System Manager パースペクティブの「統合コンポーネント・ライブラリー」の下で、自分が作成した統合コンポーネント・ライブラリー・プロジェクトのフォルダーを展開します。
2. フォルダー「ビジネス・オブジェクト」を選択します。
3. System Manager メニューから、「ファイル」>「インポート」を選択します。
4. 「インポート (選択) (Import (Select))」ダイアログが表示されます。「ファイル・システム」を選択し、「次へ」を選択します。
5. 「インポート (ファイル・システム)」ダイアログが表示されます。使用するビジネス・オブジェクト定義が *.xsd ファイルの形式で格納されているフォルダーに、「参照」メニューを使用して移動します。
6. 使用するファイルのボックスにチェックマークを付けます。デフォルトでは「*選択済みのフォルダーのみ作成する (Create selected folders only)*」にチェックマークが付いているので、そのままにします。「完了 (Finish)」を選択します。
7. ダイアログが閉じたら、自分が作成した統合コンポーネント・ライブラリー・プロジェクトのフォルダーにある「ビジネス・オブジェクト」フォルダーをもう一

度選択します。右マウス・ボタンでクリックし、「最新表示」を選択します。インポートしたビジネス・オブジェクト定義が、「ビジネス・オブジェクト」フォルダー内に表示されます。

WebSphere Application Server 用の WebSphere MQ キューの構成

WebSphere Application Server がコネクターと連動できるようにするには、WebSphere MQ キューを作成して構成する必要があります。

WebSphere Application Server 環境では、物理的な WebSphere MQ キューの名前を、WebSphere Application Server のバインディング・プロパティーに使用される JNDI 名にマップすることができます。後で事情が変わって別のキューまたはキュー・マネージャーを使用しなければならなくなった場合には、ローカル構成ファイル内でキューまたはキュー・マネージャーの名前を変更し、さらに WebSphere Application Server の管理コンソールでもその名前を変更すれば、.wsdl ファイルを再生成する必要がありません。

キューを作成および構成するための作業は、以下のとおりです。

- 実装しようとしている対話パターンに対応する WebSphere MQ キューを作成する。この作業については、60 ページの『WebSphere MQ キューの作成』で説明します。
- コネクターの構成プロパティーに、キュー・マネージャーおよびキューの名前を指定する。この作業については、 75 ページの『構成の ICL としての保管』で説明します。
- コネクター構成ファイルを WebSphere Application Server プロジェクト内にエクスポートし、プロジェクト・ファイル内のキュー名を構成して、構成ファイルから取得される物理キュー名が WebSphere Application Server の管理コンソールで指定する JNDI 名と関連付けられるようにする。この作業については、 78 ページの『ユーザー・プロジェクトの配置』で説明します。
- WebSphere Application Server の管理コンソールで、各キューに対応する JNDI 名を設定する。この作業については、第 9 章で説明します。

43 ページの『メッセージ・キュー』では、WebSphere Business Integration システムで WebSphere MQ キューがどのように使用されるかについて説明しています。 68 ページの『キュー・マネージャーによる接続モードの設定』では、コネクターの構成ファイルに接続モードを指定する方法について説明しています。

WebSphere MQ キューの作成

ビジネス・インテグレーション・システムでは、下記のプロパティーを持つキューを作成する必要があります。

注: 68 ページの『コネクターが使用するキューの指定』でコネクターを構成する際、各キューの名前をコネクターの標準プロパティーとしてコネクターの構成ファイルに指定する必要があります。

- **DeliveryQueue:** コネクター・フレームワークから統合ブローカーにイベント・デリバリー・メッセージを配信します。

- **RequestQueue:** 統合ブローカーからコネクター・フレームワークに Hub One Way (ハブ片方向) 要求メッセージまたは Hub Request (ハブ要求) 要求メッセージを配信します。
- **ResponseQueue:** ハブ要求への応答時に、コネクター・フレームワークから統合ブローカーに応答メッセージを配信します。
- **FaultQueue:** コネクター・フレームワークから統合ブローカーに障害メッセージを配信します。コネクター・フレームワークは、返信先キューにメッセージを配置することができないとき、このキューにメッセージを配置します。
- **SynchronousRequestQueue:** コネクター・フレームワークから統合ブローカーに要求メッセージを配信します。
- **SynchronousResponseQueue:** SynchronousRequestQueue への応答時に、統合ブローカーからコネクター・フレームワークへ応答メッセージを配信します。
- **AdminInQueue:** 統合ブローカーからコネクター・フレームワークに管理メッセージを配信します。
- **AdminOutQueue:** コネクター・フレームワークから統合ブローカーに管理メッセージを配信します。

キューを定義する方法

次のいずれかの方法を使用して、ご使用のアダプターのために必要な WebSphere MQ キューを構成することができます。

- WebSphere Business Integration Adapters に用意されているバッチ・ファイルをカスタマイズし、実行する。
- WebSphere MQ Explorer を使用する。
- WebSphere MQ コマンドを実行する。

ヒント

キューが関連付けられているコネクターを簡単に識別するには、コネクター名をキュー名の接頭部として使用します。例えば、Clarify コネクターのイベント・デリバリー・キューの名前は次のようにします。

```
clarifyconnector/deliveryqueue
```

WebSphere Business Integration Adapter バッチ・ファイルを使用して WebSphere MQ キューを構成する方法: WebSphere Business Integration Adapters には、1 セットのバッチ・ファイルが用意されていおり、それらのバッチ・ファイルを使用して、展開するアダプターに必要な WebSphere MQ キューを構成することができます。*ProductDir* ¥templates にあるバッチ・ファイルは、以下のファイルから構成されています。

- **configure_mq.bat** (Windows)
  configure_mq (UNIX)

  このバッチ・ファイルを実行して、crossworlds_mq.tst に指定されている WebSphere MQ キューを構成します。
- **crossworlds_mq.tst**
  このファイルを編集して、ビジネス・インテグレーション・システムの

WebSphere MQ キューを指定します。このファイルは、ご使用の WebSphere Business Integration Adapter とともに提供された、WebSphere MQ キューからのメッセージをクリアするためのバッチ・ファイル、configure_mq.bat および clear_mq.bat によって入力データとして読み取られます。

clear_mq.bat の使用についての詳細は、121 ページの『WebSphere MQ キューからのメッセージのクリア』を参照してください。

crossworlds_mq.tst ファイルの内容を以下に示します。 この 1 つのファイルを使用して、構成する各アダプターで必要なキューを指定できます。このファイルは、次のように編集してください。

1. 以下のステートメントを削除します。
   DEFINE QLOCAL(IC/SERVER_NAME/DestinationAdapter)
   DEFINE QLOCAL(AP/DestinationAdapter/SERVER_NAME) これらのステートメントは、WebSphere InterChange Server を使用するビジネス・インテグレーション・システムにのみ適用されます。
2. DEFINE QLOCAL(*AdapterName*/AdminInQueue) から始まるステートメントをテンプレートとして使用して、配置するアダプターごとに個別のキュー定義ステートメント・セットを作成します。
3. リモート・キュー定義を指定しバインディング・モードを使用している場合は、構成する必要のある各キュー・マネージャーごとに、要求された情報を使用して、ステートメント DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN) TRPTYPE(TCP) をカスタマイズしてください。キュー構成にクライアント・モードを使用している場合には、ステートメントは現状のままにしてください。サポートされているキュー構成の詳細については、63 ページの『キュー構成の定義』を参照してください。

```
*************************************************************/
*   For each JMS queue (delivery Transport is JMS),
*   default values follow the convention:
*           AdapterName/QueueName
*************************************************************/
    DEFINE QLOCAL(AdapterName/AdminInQueue)
    DEFINE QLOCAL(AdapterName/AdminOutQueue)
    DEFINE QLOCAL(AdapterName/DeliveryQueue)
    DEFINE QLOCAL(AdapterName/RequestQueue)
    DEFINE QLOCAL(AdapterName/ResponseQueue)
    DEFINE QLOCAL(AdapterName/FaultQueue)
    DEFINE QLOCAL(AdapterName/SynchronousRequestQueue)
    DEFINE QLOCAL(AdapterName/SynchronousResponseQueue)
*************************************************************/
*   Define the default CrossWorlds channel type             */
*************************************************************/
    DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN) TRPTYPE(TCP)
*************************************************************/
*   End of CrossWorlds MQSeries Object Definitions          */
*************************************************************/
```

WebSphere MQ Explorer を使用して WebSphere MQ キューを構成する方法:

WebSphere MQ Explorer を使用したキューの構成については、WebSphere MQ Explorer を開いて、そのオンライン・ヘルプを参照してください。

WebSphere MQ コマンドを使用して WebSphere MQ キューを構成する方法:
WebSphere MQ コマンドを使用してキューを構成する方法の詳細は、「*WebSphere MQ: システム管理ガイド*」および「*WebSphere MQ: Script MQSC コマンド・リファレンス*」を参照してください。

キュー構成の定義

WebSphere Business Integration システムでは、いくつかのキュー・マネージャーとキュー構成をサポートしています。コネクターは、以下のどのモードにおいても、キュー・マネージャーと通信することができます。

バインディング・モード

バインディング・モードでは、統合ブローカーとコネクターはキュー・マネージャーを使用して直接通信することが可能で、TCP/IP 接続は使用しません。統合ブローカーとコネクターは、同じキュー・マネージャーを共用できるように、同一のマシンにインストールする必要があります。デフォルトでは、このバインディング・モードになります。

リモート・キュー定義でのバインディング・モード

WebSphere Application Server とコネクターが別々のマシンにインストールされ、各マシンで独自のキュー・マネージャーが動作している場合、コネクターと統合ブローカーは、バインディング・モードを使ってそれぞれのキュー・マネージャーとそのまま通信することができます。ただし、次の例で説明するとおりに、リモート・キュー定義を指定する必要があります。

例えば、brokerQM は統合ブローカーが使用するキュー・マネージャーで、connQM はコネクターが使用するキュー・マネージャーであるとします。双方のキュー・マネージャー間で通信ができるようにするには、次のチャネル定義をセットアップする必要があります。

- コネクターからブローカーにメッセージを転送する各キューでは、brokerQM のローカル・キューを指すリモート・キュー定義を connQM に作成する必要があります。この要件は次のキューに適用されます。
  - DeliveryQueue
  - SynchronousRequestQueue
  - FaultQueue
  - AdminOutQueue
- 統合ブローカーからコネクターにメッセージを転送する各キューでは、connQM のローカル・キューを指すリモート・キュー定義を brokerQM に作成する必要があります。この要件は次のキューに適用されます。
  - RequestQueue
  - AdminInQueue
- 応答先キューの場合 : 統合ブローカーがコネクター・フレームワークに要求メッセージを送信するときに、メッセージ・ヘッダーにキュー・マネージャーおよび応答の送信先となるキューを指定します。これは、コネクター・フレームワークから統合ブローカーに送信される要求にも当てはまります。特定の管理用タスク

を実行して、応答メッセージが正しい応答先キューへ送信されるようにする必要があります。これらについては、「*WebSphere MQ: 相互通信*」を参照してください。

クライアント・モード

WebSphere Application Server とコネクターがそれぞれのキュー・マネージャーと通信するために TCP/IP を使用する必要がある場合は、クライアント・モード接続を使用しなければなりません。通信は、基本となるトランスポート層として TCP/IP を使用するクライアント接続を介して行われます。

アプリケーションでコネクターを使用可能にする

コネクターのアプリケーション固有のコンポーネントが、アプリケーションとの間で業務データを送受信できるようにするには、アプリケーションのコネクターに専用の、ユーザー・アカウントを設定する必要があります。コネクターの構成ファイルを作成する際には、このアカウントのユーザー ID とパスワードを指定する必要があります。

ほとんどのコネクターに関して、アプリケーションはイベント検出機構を実装するように構成されている必要があります。構成が完了したアプリケーションは、エンティティーの変更を検出してイベント・レコードをイベント・ストアに書き込むことができます。その後は、コネクターによって取り出され、処理されます。トリガーの作成は、WebSphere MQ メッセージ・フローで処理される、ビジネス・オブジェクト用とオペレーション用に限定します。そうしないと、処理にともない除去の対象となることのないメッセージで、メッセージ・キューが満杯の状態になります。

アプリケーションがコネクターと連動できるようにするために必要な全作業に関する詳細については、構成するコネクターのアダプター・ユーザー・ガイドを参照してください。

コネクターの構成

Connector Configurator ツールには、コネクターを構成するためのグラフィカル・ユーザー・インターフェース (GUI) が用意されています。コネクターの構成プロパティーの値を指定する作業が完了すると、Connector Configurator によってコネクターの構成ファイルが生成されます。

Connector Configurator を使用して作成したコネクター構成は、WebSphere Application Server の対話パターンに適したサービス・プロジェクトを生成する際に使用する .wsdl ファイルのソースとして利用されます。また、このコネクター構成は、コネクター・エージェント (アダプター・エージェントとも呼ばれます) のインストール先と同じ場所にインストールされて、ローカル構成ファイルとしても使用されます。構成ファイルを作成するステップを完了すると、System Manager パースペクティブで、構成を統合コンポーネント・ライブラリー・プロジェクトとして保管できるようになります (これにより、*.wsdl ファイルの生成にその構成を使用できるようになります)。また、コネクター・エージェントがインストールされている場所に、そのエージェントに関連するファイルとして構成を保管できるようになります。

重要: ビジネス・インテグレーション・アダプターが UNIX 上で動作する場合、Windows で Connector Configurator を使用して構成ファイルを作成し、そのファイルを自分の UNIX マシンにコピーする必要があります。構成ファイルを作成する場合、プロパティーを設定する際に UNIX パスとファイル名の命名規則を守ってください。

購入したアダプター製品によっては、コネクター構成の際に参考にできるファイルが用意されていることがあります。このファイルは、ディレクトリー *ProductDir*¥connectors¥repository¥*ConnName* 内のコネクター定義ファイル、またはサンプル構成ファイルです。これらのファイルを Connector Configurator で開いて修正することにより、そのコネクターに適した完全な構成ファイルを作成できます。コネクター定義ファイルは構成ファイルの作成時に参考となる値をいくつか示すものであるのに対し、完成した構成ファイルは、コネクターの標準およびアプリケーション固有のプロパティーをすべて含み、サポートされるビジネス・オブジェクトも指定したものであることに注意してください。これらのプロパティーの詳細は、145 ページの『付録 B. コネクターの標準構成プロパティー』に説明されています。

コネクター定義ファイルやサンプル構成ファイルが提供されていない場合には、Connector Configurator を使用して、コネクターの構成ファイルを新しく作成する必要があります。

以降の各セクションでは、WebSphere Application Server と連動するすべてのコネクターに適用される Connector Configurator の設定について説明します。

Connector Configurator の実行

新しいコネクター構成を作成したり、System Manager の統合コンポーネント・ライブラリー内に保管されている既存のコネクター構成を変更したり、System Manager の外部にある既存のコネクター構成ファイルを開いたりするには、Connector Configurator を使用します。

WebSphere Application Server を統合ブローカーとするコネクター構成を使用するには、どの方法で構成プロセスを開始したかに関係なく、完成したコネクター構成を System Manager の統合コンポーネント・ライブラリーに含まれるプロジェクトに保管する必要があります。

Connector Configurator を始動して実行するには、次のいずれかの手順に従います。

コネクター構成を新しく作成する場合は、以下の手順を実行します。

1. System Manager パースペクティブを開きます。
2. System Manager パースペクティブで、「統合コンポーネント・ライブラリー」フォルダーを展開し、既存の統合コンポーネント・ライブラリー・プロジェクトのフォルダーを選択します。(プロジェクト・フォルダーがまだない場合は、新しく作成する必要があります。統合コンポーネント・ライブラリー・プロジェクトのフォルダーを新しく作成するには、「統合コンポーネント・ライブラリー」を選択し、右マウス・ボタンでクリックして、「新規統合コンポーネント・ライブラリー」ダイアログを開きます。新しく作成するライブラリー・プロジェクトのプロジェクト名を入力し、「完了 (Finish)」をクリックします。トップレベルの「統合コンポーネント・ライブラリー」フォルダー内に、新しく作成したライ

ブラリー・プロジェクトがフォルダーとして表示されます。) 使用する統合コンポーネント・ライブラリー・プロジェクトのフォルダーを展開し、その中の「コネクター」を選択します。

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. 新しく作成するコネクター構成の名前を入力します。
5. 「システム接続: 統合ブローカー (System Connectivity: Integration Broker)」の横のプルダウン・メニューをクリックし、 「WAS」を選択します。
6. テンプレートが存在する場合は、選択します。テンプレートがない場合は、「なし」を選択してください。
7. 構成ファイルの設定を行う前に、「標準のプロパティー」タブをクリックして、次のように設定されていることを確認します。
   - BrokerType は WAS です。
   - DeliveryTransport は JMS です
   - WireFormat は CwXML です

以前に System Manager 内に保管した既存の構成を編集する場合は、以下の手順を実行します。

1. System Manager パースペクティブで、「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。
2. 「コネクター」フォルダー内に表示されたコネクター構成のリストから 1 つを選択し、右マウス・ボタンでクリックして、「定義の編集」を選択します。Connector Configurator が開き、コネクターの構成が、統合ブローカーのタイプおよびファイル名を先頭にして表示されます。
3. 構成ファイルの設定を行う前に、「標準のプロパティー」タブをクリックして、次のように設定されていることを確認します。
   - BrokerType は WAS です。
   - DeliveryTransport は JMS です
   - WireFormat は CwXML です

既存のファイルを使用してコネクターを構成する場合は、そのファイルを Connector Configurator 内で開き、構成を変更してから、System Manager 内の統合コンポーネント・ライブラリーにその構成を保管する必要があります。

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

1. Connector Configurator 内で、「ファイル」>「開く」>「ファイルから」をクリックします。
2. 「ファイル・コネクターを開く」ダイアログ内で、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。
4. 構成ファイルの設定を行う前に、「標準のプロパティー」タブをクリックして、次のように設定されていることを確認します。
   - BrokerType は WAS です。
   - DeliveryTransport は JMS です
   - WireFormat は CwXML です

上記の説明に従って Connector Configurator を開いた後は、このセクションの説明に従って、プロパティー (キューを指定するプロパティーを含みます) を設定し、サポートされるビジネス・オブジェクトを指定します。構成が完了したら、「プロジェクトに保管」を選択してください。

標準およびコネクター固有のプロパティーの設定

コネクターを構成するときには、標準のプロパティーとコネクター固有のプロパティーの両方に値を設定する必要があります。コネクター固有のプロパティーについては、使用するアダプターの資料を参照してください。標準のプロパティーについては、本書の 145 ページの『付録 B. 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。

以下のプロパティー設定は、WebSphere Application Server を統合ブローカーとして使用するコネクターでは必須の設定です。

- BrokerType は WAS に設定されていなければなりません
- DeliveryTransport は JMS に設定されていなければなりません
- WireFormat は CwXML に設定されていなければなりません

サポートされるビジネス・オブジェクトの指定

ビジネス・オブジェクトをコネクターでサポートされるビジネス・オブジェクトに指定するには、そのビジネス・オブジェクトを、統合コンポーネント・ライブラリーのフォルダーに含まれるプロジェクトにあらかじめ保管しておく必要があります(59 ページの『ICL へのビジネス・オブジェクトの追加』を参照)。以下の説明は、この保管がすでに行われていることを前提としています。

既存のビジネス・オブジェクトをコネクターでサポートされるビジネス・オブジェクトに指定するには、以下の手順を実行します。

1. Connector Configurator でコネクター構成を開きます。「サポートされているビジネス・オブジェクト」タブを選択します。

2. 「ビジネス・オブジェクト名」ドロップダウン・ボックスが表示されます。現在のコネクター構成でサポートされるビジネス・オブジェクトが指定されていない場合、最初はブランクのフィールドが表示されます。ドロップダウン矢印をクリックします。
3. スクロール・リストが表示されます。このリストには、現在のコネクター構成でサポートされるビジネス・オブジェクトに指定できるビジネス・オブジェクトが、すべて表示されます。このリストにビジネス・オブジェクトが表示されるようにするには、現在のコネクター構成が含まれている「コネクター」サブフォルダーの格納先と同じ統合コンポーネント・ライブラリー・プロジェクトの「ビジネス・オブジェクト」サブフォルダーに、表示するビジネス・オブジェクトをあらかじめ保管しておかなければなりません。
4. ドロップダウン・フィールドでリストに表示されたビジネス・オブジェクトの 1 つをクリックすることにより、そのビジネス・オブジェクトを現在のコネクター構成でサポートされるビジネス・オブジェクトに指定します。そのビジネス・オブジェクトの名前が最初のドロップダウン・フィールドに入力されます。また、新しいブランク・フィールドが追加されます。新しいブランク・フィールドで同じ操作を行います。必要なビジネス・オブジェクトがすべて指定されるまで、この操作を繰り返します。
5. 完了したら、「ファイル」>「プロジェクトに保管」を選択します。これにより、コネクター構成が統合コンポーネント・ライブラリー・プロジェクトとして保管され、対応するアイコンが System Manager パースペクティブに表示されます。

コネクターが使用するキューの指定

60 ページの『WebSphere MQ キューの作成』では、コネクターが WebSphere Application Server との通信に使用するキューのセットを定義しました。Connector Configurator で、**「標準のプロパティー」**タブをクリックし、次の標準プロパティーに値を設定することにより、これらのキューをコネクターに割り当てます。

- DeliveryQueue
- RequestQueue
- ResponseQueue
- FaultQueue
- SynchronousRequestQueue
- SynchronousResponseQueue
- SynchronousRequestTimeout
- AdminInQueue
- AdminOutQueue

キュー・マネージャーによる接続モードの設定

デフォルトの接続モードはバインディング・モードです。クライアント・モードは、次のように指定してください。

1. Connector Configurator で、**「標準のプロパティー」**タブをクリックします。

2. 標準プロパティー jms.MessageBrokerName に次の値を割り当てます:
QueueMgrName:[*Channel*]:[*HostName*]:[*PortNumber*]。ここで、変数はそれぞれ以下のものを表します。

QueueMgrName
キュー・マネージャーの名前。

Channel
クライアントが使用するチャネル。

HostName
キュー・マネージャーが常駐するマシンの名前。

PortNumber
キュー・マネージャーが listen に使用するポート番号。

これには、次のものがあります。

```
jms.MessageBrokerName = WMQIB.Queue.Manager:CHANNEL1:RemoteMachine:1456
```

ロギングおよびトレース・オプションの構成

ロギングは、システム・メッセージ、コンポーネント状態変更、障害、およびトレース情報の通信に使用されます。次のファイルが生成されます。

表 8. コネクターのロギングおよびトレース・ファイル

デフォルトのファイル名およびパス	説明
一時的ログ・ファイル: *ProductDir*¥*broker_name_connector_name*_tmp.log.	始動時に、コネクターにより一時的ログ・ファイルが生成されます。このファイルには、コネクター・フレームワークに渡されるコネクター・プロパティーおよびビジネス・オブジェクト定義など、始動時にログとして記録されるメッセージがすべて格納されます。また、ファイルは製品がインストールされているディレクトリーに書き込まれます。
コネクター・ログ・ファイル: **UNIX:** メッセージはコネクターにより、デフォルトでは STDOUT にロギングされ、その後 *ProductDir*/logs/connector_manager_*ConnName*に転送されます。 **WINDOWS:** メッセージはコネクターによりデフォルトでは STDOUT にロギングされますが、*ProductDir* ディレクトリー内のローカル宛先ログ・ファイルに送信するように構成することもできます。	コネクターのログ・ファイルは、コネクターから発行されたメッセージを保管するために使用します。また、WebSphere MQ 通信エラーに関するメッセージも格納されます。
コネクター・メッセージ・ファイル: *ProductDir* ¥connectors¥messages¥*ConnName_LocaleName* .txt	このファイルには、コネクターから発行される各メッセージのテキスト全文が格納されます。このファイルを使用して、ログ・ファイルに表示されるメッセージ ID の本文を検索することができます。コネクター構成ファイルで指定したロケールがサポートされていない場合は、ファイル *ConnName*.txt が使用されます。
トレース・ファイル: UNIX と Windows の双方とも、デフォルトでは STDOUT に設定されています。	選択したトレース・レベルで指定されたトレース・メッセージが格納されます。

ロギング・システムは常にアクティブで、コネクターの正確なモニター情報が出力されます。

障害発生時の対処では、トレースをオンにします。トレース・メッセージにより、ビジネス・インテグレーション・システムのコンポーネントでの動作状況をモニターすることができます。トレース・レベルにより、トレース・ファイルに書き込む詳細情報の量が定義されます。トレース・レベルが高いほど、詳細な情報が書き込まれます。トレースは次の点で、ロギングとは異なります。

- ロギングは常時実行されますが、トレースは必要に応じて、オン/オフ状態にすることができます。
- トレースでは、コンポーネントの状態とコンポーネントの動作について、ロギングよりも詳細な情報が記録されます。
- ロギングとトレースの設定値は、リブート後も有効です。

トレースは、デフォルトではオフの状態になっています。これは通常必要とされる以上に詳細なメッセージを出力するからです。

LogViewer を使用したロギングおよびトレース・メッセージの表示については、130 ページの『Log Viewer を使用してコネクター・メッセージを表示する方法』を参照してください。

コネクター・ロギングの構成

コネクター・ロギング・オプションを構成するには、**「トレース/ログ・ファイル」**タブをクリックして、下記のとおりに指定します。

1. ログ・メッセージを STDOUT に転送する場合は、**「コンソールに (STDOUT)」**チェック・ボックスをクリックする。
2. ログ・メッセージをファイルに転送する場合は、**「ファイルに」**チェック・ボックスをクリックし、使用するログ・ファイルの絶対パス名を指定する。「コンソールに」と「ファイルに」の両オプションを指定して、ログ・メッセージをコンソールおよびファイルに転送することもできます。
3. ログ・ファイルの使用を指定した場合は、次のオプションも指定する。
   a. ログ・ファイルのサイズを制限するには、**「ログ・ファイルのサイズ」**に数値と測定単位を設定する。
   b. ログ・ファイルのサイズを制限なしとするには、**「無制限」**チェック・ボックスをクリックする。
   c. ログ・ファイルの最大サイズを設定し、ファイルのアーカイブを使用する場合は、**「アーカイブの数」**に保存するアーカイブ・ファイル数を設定する。

ログ・ファイル管理の詳細については、 121 ページの『ログ・ファイルおよびトレース・ファイルの管理』を参照してください。

コネクター・トレースの構成

コネクターのトレース・オプションを構成する作業は、次の手順で行います。

1. **「トレース/ログ・ファイル」**タブをクリックする。
2. トレース・メッセージを STDOUT に転送するため、**「コンソールに (STDOUT)」**にチェックマークを付ける。
3. トレース・メッセージをファイルに転送するため、**「ファイルに」**にチェックマークを付け、使用するトレース・ファイルの絶対パス名を指定する。**「コンソールに」**と**「ファイルに」**の両オプションを指定して、トレース・メッセージをコンソールおよびファイルに転送することもできます。
4. トレース・ファイルの使用を指定した場合、次のオプションも指定する。
   a. トレース・ファイルのサイズを制限するには、**「トレース・ファイルのサイズ」**に数値と測定単位を設定する。
   b. トレース・ファイルのサイズを制限なしとするには、**「無制限」**にチェックマークを付ける。
   c. トレース・ファイルの最大サイズを設定し、ファイルのアーカイブを使用する場合は、**「アーカイブの数」**に保存するアーカイブ・ファイル数を設定する。
5. トレース・レベルを設定するには、次の作業を実行します。
   a. **「標準のプロパティー」**タブをクリックする。
   b. **AgentTraceLevel** プロパティーに、 72 ページの表 9 にリストされている値のうちの 1 つを設定する。

構成するコネクターの場合に個々のトレース・レベルで生成される情報の詳細については、そのコネクターのアダプター・ユーザー・ガイドを参照してください。コネクターのトレースは、次のいずれかのレベルに設定します。

表 9. コネクター・トレース・レベル

トレース・レベル	説明
1	トレースの初期化、およびビジネス・オブジェクトの送受信。
2	レベル 1 のメッセージ印刷。さらに、同じタイプのイベントについて、レベル 1 よりも詳細な情報を出力。
3	レベル 1 および 2 のメッセージ印刷。さらに、コネクター・エージェントとメッセージング・ドライバー間におけるメッセージ交換のトレース。
4	レベル 1 から 3 のメッセージ印刷。さらに、コネクターの内部レベル間におけるビジネス・オブジェクトの引き渡し処理のトレース。
5	レベル 1 から 4 のメッセージ印刷。さらに、コネクターの内部レベル間における管理メッセージの引き渡し処理のトレース。

新規または変更済みのトレース・レベルは、コネクターを再始動すると有効になります。

トレース・ファイルをアーカイブする作業の詳細については、121 ページの『ログ・ファイルおよびトレース・ファイルの管理』を参照してください。

コネクターの始動ファイル、ショートカット、および環境変数の構成

コネクターの始動手順も、必要なセットアップ・タスクも、コネクターが動作するプラットフォームにより、以下のように異なります。

Windows の場合

Windows に WebSphere Business Integration Adapters をインストールすると、インストールされた各コネクターについて WebSphere Business Integration Adapters プログラム・メニュー (「スタート」>「プログラム」>「IBM WebSphere Business Integration Adapters」>「アダプター」>「コネクター」) にショートカットが作成されます。下記のものをカスタマイズするには、163 ページの『付録 C. コネクター始動オプション』にリストされている始動オプションを使用する必要があります。

- コネクターのショートカット・プロパティー
- コネクターの始動ファイル `start_connName.bat` (Java で作成されたコネクターの場合)。
- コネクターの始動ファイル `start_connector.bat` (C++ で作成されたコネクターの場合)。

UNIX の場合

UNIX 環境では、connector_manager_connName スクリプトを実行してコネクターを始動します。このスクリプトは、汎用コネクター・マネージャー・スクリプト (*ProductDir*/bin/connector_manager) のラッパーです。このラッパーには、以下の情報が組み込まれています。

- 始動または停止するコネクター名。
- 汎用コネクター・マネージャーに対応するコマンド行オプション。これには、次のものがあります。
  - SAP コネクターには -t コマンド行オプションが必要です。したがって、その始動スクリプトには -t オプションが組み込まれます。
  - UNIX コネクターはすべて -b オプションを使用して動作します。したがって、コネクター始動スクリプトすべてに -b オプションが組み込まれます。コネクターをフォアグラウンドで実行するには、汎用コネクター・マネージャー・スクリプト (connector_manager) から -b オプションを除去します。
- 構成ファイルの名前。

カスタム・アダプターを作成した場合や、アダプターを ESD (Electronic Software Delivery) を使用してインストールした場合は、コネクターを初めて始動する際に、以下の手順を行なう必要があります。

1. コネクター・スクリプト生成ツールを実行し、connector_manager_*connName* スクリプトを、コネクターの構成ファイルの名前で更新する。このツールの実行の詳細については、167 ページの『付録 D. コネクター・スクリプト生成ツールの使用』を参照してください。

   あるいは、*ProductDir*/bin ディレクトリーにナビゲートして connector_manager_*connName* ファイルを編集し、コネクターの構成ファイル名を指定する。ファイル内で、AGENTCONFIG_FILE 変数を検索し、その変数に次のように構成ファイルの絶対パス名を設定する。

   ```
   AGENTCONFIG_FILE=ConfigFile
   ```

2. 必要であれば、PATH 環境変数を更新して *ProductDir*/bin ディレクトリーを組み込む。
3. CWSharedEnv.sh ファイルは、自分のアカウントのシェル始動スクリプト (.cshrc など) が、ソースになっていることを確認する。

始動スクリプトのカスタマイズ: 汎用コネクター・マネージャー・スクリプトは、特定のコネクターを管理する実スクリプトである、該当の start_ *connector*.sh スクリプトを呼び出します。各 WebSphere Business Integration Adapter には、start_*connector*.sh スクリプトが含まれています。start_*connector*.sh スクリプトを変更して、163 ページの『付録 C. コネクター始動オプション』で紹介されている各サポート始動オプションを、任意に組み込むことができます。

注: コネクター用の始動ファイルの作成については、「*コネクター開発ガイド (C++ 用)*」または「*コネクター開発ガイド (Java 用)*」を参照してください。

Visual Test Connector を使用したインターフェースの検証

Visual Test Connector は、コネクターの活動をシミュレートすることで、実際にコネクターを実行するという手間をかけずに、統合コンポーネントをテストできます。Visual Test Connector を使用して、ソース・コネクターおよび宛先コネクターが正しく構成されているかどうか、およびサポートされているビジネス・オブジェクト定義が正しく指定されているかどうかを検証できます。Visual Test Connector の使用方法についての詳細は、203 ページの『付録 F. Visual Test Connector の使用』を参照してください。

第 8 章 WebSphere Application Server への配置

本章の作業を実行する前に、以下の作業を完了しておく必要があります。

- 実装するデータ・トランザクションに適した JMS キューを作成し構成する。
- Connector Configurator を使用して、関連するビジネス・オブジェクトをサポートするようにコネクターを構成する。

本章では、以下の作業の実行方法について説明します。

1. コネクター構成および関連するビジネス・オブジェクトを、System Manager 内で統合コンポーネント・ライブラリーとして保管する。
2. WebSphere Studio Application Developer Integration Edition の System Manager パースペクティブ内で、統合コンポーネント・ライブラリーのグループから成るユーザー・プロジェクトを作成する。
3. System Manager のユーザー・プロジェクトを、WebSphere Studio Application Developer Integration Edition の System Manager パースペクティブから Business Integration パースペクティブに配置する。
4. Business Integration パースペクティブを使用して、各対話パターン用の WebSphere Application Server コンポーネントを作成する。また、WebSphere Application Server Administrative の管理コンソールを使用して、キューのセットアップを完了する。
5. 対話パターンごとに、サーバーに配置可能な Enterprise Application Archive プロジェクトを作成する。

構成の ICL としての保管

WebSphere Business Integration Adapters では、構成済みのリソース (統合コンポーネントと呼びます) が使用されます。これらのリソースは、System Manager パースペクティブを通じて管理されます。統合コンポーネントは、統合コンポーネント・ライブラリー (ICL) と呼ばれるセットにまとめられます。すべてのアダプターで広く使用されている統合コンポーネントは、コネクターとビジネス・オブジェクト定義です。

アダプターを実装するには、コネクターを構成し、そのコネクターでサポートされるビジネス・オブジェクト定義を指定する必要があります。これらのビジネス・オブジェクト定義とコネクター構成は、両方とも System Manager 内の統合コンポーネント・ライブラリー・プロジェクトに保管する必要があります。これを行うには、以下のステップを実行します。

1. System Manager パースペクティブで、統合コンポーネント・ライブラリー・パースペクティブを右マウス・ボタンでクリックし、「新規統合コンポーネント・ライブラリー」を選択します。
2. 「新規統合コンポーネント・ライブラリー」ダイアログで、「プロジェクト名」フィールドにライブラリー名を指定し、「終了」を選択します。指定した名前を持つ新しいフォルダーが、統合コンポーネント・ライブラリーに追加されます。このフォルダーには、いくつかのタイプの統合コンポーネントに対応

するサブフォルダーが格納されています。ただし、WebSphere Application Server をブローカーとしてコネクターを実装する場合、必要なのは「ビジネス・オブジェクト」フォルダーと「コネクター」フォルダーだけです。

3. 「ビジネス・オブジェクト」フォルダーを選択し、「ファイル」>「インポート」を選択します。「インポート」ダイアログが表示されます。
4. 「ファイル・システム」を選択し、「次へ」を選択します。「インポート (ファイル・システム)」ダイアログが表示されます。「ディレクトリー」ドロップダウン・リストで、作成済みの対話パターンでコネクターと共に使用するビジネス・オブジェクト定義ファイル (.xsd) が格納されているフォルダーを参照します。
5. フォルダーを選択し、「OK」を選択します。「インポート」ダイアログが表示されます。このダイアログでは、選択可能なリソースのそれぞれの横にチェック・ボックスが表示されます。使用する .xsd ファイルのそれぞれのチェック・ボックスにチェックマークを付け、「完了 (Finish)」を選択します。
6. System Manager の「統合コンポーネント・ライブラリー」の下にある「ビジネス・オブジェクト」フォルダーで、「ビジネス・オブジェクト」フォルダーを右マウス・ボタンでクリックし、表示を最新にします。インポートされたビジネス・オブジェクト定義を表すアイコンが、このフォルダーに表示されます。
7. 「コネクター」フォルダーを選択し、「ツール」メニューから「Connector Configurator」を選択します。Connector Configurator ツールがアクティブになり、「新規コネクター」ダイアログが表示されます。
8. Connector Configurator ツールを使用して、構成プロパティーの設定と、コネクターでサポートされるビジネス・オブジェクトの指定を行います。コネクター構成ファイルのサンプルか、部分的に構成済みのコネクター構成ファイルがある場合には、「新規コネクター」ダイアログを取り消します。次に、「Connector Configurator」ダイアログで、「ファイル」>「開く」>「ファイルから」を選択します。「ファイル・コネクターを開く」ダイアログが表示されます。コネクター構成ファイルの既存のサンプルまたは部分的に構成済みの既存のコネクター構成ファイルの保管先に移動し、そのファイルを選択して「開く」を選択します。Connector Configurator のプロパティー画面が表示されます。この画面には、選択したコネクター構成を示す名前が付いています。
9. 新規のコネクター構成を最初から作成する場合には、統合ブローカーに WAS を選択し、この新規コネクター構成の名前を指定します。次に、プロパティー・テンプレートを選択し (プロパティー・テンプレートがまだない場合は「なし」を選択します)、「OK」をクリックします。Connector Configurator のプロパティー画面が表示されます。この画面には、作成される新規コネクター構成を示す名前が付いています。
10. WebSphere Application Server と共に使用するコネクターについては、必ず以下のタブで構成値を設定する必要があります。
    a. 「標準のプロパティー」タブで、統合ブローカーを使用するために必要な値 (対話パターンの実装に使用されるキューを含みます) を設定します。68 ページの『コネクターが使用するキューの指定』および 60 ページの『WebSphere Application Server 用の WebSphere MQ キューの構成』を参照してください。

b. 「サポートされているビジネス・オブジェクト」タブで、実装するコネクターと共に使用するためにサポートするビジネス・オブジェクトを指定します。67 ページの『サポートされるビジネス・オブジェクトの指定』を参照してください。

c. 各タブでの値の設定が完了したら、「ファイル」>「別名保管」>「プロジェクトに」を選択します。「別名保管」ダイアログが表示されます。現在のコネクター構成を保管する統合コンポーネント・ライブラリーを選択し、コネクター構成名として使用する名前を入力して、「保管」を選択します。Connector Configurator によってコネクター構成の検証が行われ、検証の成功または失敗を通知するメッセージが表示されます。現在のコネクターの構成プロパティーに加える変更がほかになければ、Connector Configurator を閉じます。

11. System Manager パースペクティブ内の自分が作成した統合コンポーネント・ライブラリー・プロジェクトの下にある「コネクター」フォルダーに、保管したコネクター構成を表すアイコンが表示されます。

12. 保管したコネクターは、指定したプロパティーおよびサポートされるビジネス・オブジェクトと共に、統合コンポーネント・ライブラリーの一部として構成されています。この構成に変更を加える必要がある場合には、対応するコネクター・アイコンをダブルクリックして、構成ファイルを Connector Configurator で開き、変更を行います。この構成に、新しいビジネス・オブジェクトをサポートされるビジネス・オブジェクトとして追加する必要がある場合には、そのビジネス・オブジェクトを、コネクターの保管先と同じ統合コンポーネント・ライブラリーにあらかじめ保管しておく必要があります。その後で Connector Configurator を開き、「サポートされているビジネス・オブジェクト」タブを選択して「ビジネス・オブジェクト名」ドロップダウン・ボックスをクリックすると、追加した新しいビジネス・オブジェクトが選択可能なビジネス・オブジェクトとして表示されます。

System Manager でのユーザー・プロジェクトの作成

System Manager のユーザー・プロジェクトは、特定の対話パターンなどの特定の用途に使用される統合コンポーネントを指定します。System Manager のユーザー・プロジェクトを作成し、その中に統合コンポーネントを保管する作業が完了すると、そのユーザー・プロジェクトを Business Integration パースペクティブに配置して、WebSphere Application Server Enterprise Application Archive プロジェクトの作成に使用することができます。

ユーザー・プロジェクトは、System Manager パースペクティブ内で「新規ユーザー・プロジェクト」ダイアログを使用して作成します。ユーザー・プロジェクトを作成するには、System Manager パースペクティブを開いて以下の手順を実行します。

1. 「ユーザー・プロジェクト」フォルダーの下で、「WAS プロジェクト」を選択して右マウス・ボタンでクリックします。
2. ポップアップ・メニューで、「新規ユーザー・プロジェクト」を選択します。「新規ユーザー・プロジェクト」ダイアログが表示されます。「プロジェクト名」フィールドに、作成するユーザー・プロジェクトの名前として使用する名前を入力します。

このダイアログに「*作成するユーザー・プロジェクトのタイプを選択してください。*」 というプロンプトが表示されている場合は、「*WebSphere Application Server プロジェクト*」 を選択します。(WebSphere Business Integration Adapters のインストール時の構成によっては、選択肢が 「*WebSphere Application Server プロジェクト*」 のみの場合があります。) 「使用可能な統合コンポーネント・ライブラリー」フィールドに、作成済みの統合コンポーネント・ライブラリーを表すフォルダーが表示されます。また、各ライブラリーとそのライブラリーに含まれる各統合コンポーネントの両方に、チェック・ボックスが表示されます。各フォルダーを展開し、作成中のユーザー・プロジェクトで使用できるようにする必要があるコンポーネントのすべてにチェックマークを付けます。「完了 (Finish)」を選択します。(統合コンポーネント・ライブラリーがまだ作成されていない場合、何も表示されません。ただし、その場合もユーザー・プロジェクトを作成することはできます。作成したユーザー・プロジェクトを後から右マウス・ボタンでクリックして「*プロジェクトを更新*」を選択すれば、その時点までに新しく作成された統合コンポーネント・ライブラリーを追加することができます。)

注: 「新規ユーザー・プロジェクト」ダイアログには、Business Integration パースペクティブから「新規」>「プロジェクト」>「WebSphere Business Integration」>「ユーザー・プロジェクト」を選択してアクセスすることもできます。新しいユーザー・プロジェクトを作成すると、System Manager パースペクティブが自動的に開きます (新しいユーザー・プロジェクトは、System Manager パースペクティブ内に作成されます)。この方法を使用しても、ここで説明しているタスクのほかの手順は変わりません。

3. ユーザー・プロジェクトが作成され、System Manager パースペクティブの「WAS プロジェクト」フォルダーの下に表示されます。

ユーザー・プロジェクトの配置

ユーザー・プロジェクトを作成して統合コンポーネント・ライブラリーをそのプロジェクトに追加した後は、そのプロジェクトを配置すれば WebSphere Application Development Environment, Integration Edition での WebSphere Application Server プロジェクト作成に使用可能なファイルを生成することができます。これを行うには、以下のステップを実行します。

1. System Manager パースペクティブで、使用するユーザー・プロジェクトを表すアイコンを右マウス・ボタンでクリックします (このとき、「統合コンポーネント・ライブラリー」フォルダー内ではなく「ユーザー・プロジェクト」フォルダー内にいることを確認してください)。ポップアップ・メニューが表示されます。
2. 「WAS プロジェクトを配置」を選択します。「コンポーネント選択ページ」が表示されます。このページでは、選択したユーザー・プロジェクトで使用可能な統合コンポーネントをリストで確認できます。(1 つのコネクターと、1 つ以上のビジネス・オブジェクトが表示されるはずです。ビジネス・オブジェクトが表示されない場合には、コネクターを構成したときにサポートされるビジネス・オブジェクトを指定したかどうか、使用するコネクター構成とビジネス・オブジェクトを統合コンポーネント・ライブラリーに追加したかどうか、および、ユーザー・プロジェクトを更新してその統合コンポーネント・ライブラリーを追加したかどうかを確認してください。) 追加するコンポーネントのボックスにチェック

マークを付け、「次へ」をクリックします。ポップアップ・メニューが表示され、次の 3 つのオプションが提示されます。

- サービス・プロジェクトへエクスポート

  コネクター構成ファイルとビジネス・オブジェクト定義ファイルが、それぞれ WSDL 形式と XSD 形式で WebSphere Studio Application Developer Integration Edition サービス・プロジェクト内に直接保管されます。保管先のサービス・プロジェクトがすでに存在しており、かつ現在のワークスペース内にあることが必要です。

- JAR ファイルとしてエクスポート

  コネクター構成ファイルとビジネス・オブジェクト定義ファイルが、それぞれ WSDL 形式と XSD 形式で JAR ファイル内に保管されます。保管先の JAR ファイルは、ローカルに使用することも、別のマシンの WebSphere Studio Application Developer Integration Edition のインストール先に移動してサービス・プロジェクトにインポートすることもできます。

- ディレクトリーへエクスポート

  ユーザー・プロジェクトのコネクター構成ファイルとビジネス・オブジェクト定義ファイルは、選択したディレクトリー・ロケーションに保管されます。コネクター構成ファイル (拡張子が .cfg または .con の XML 形式のファイル) は、WSDL (Web Services Description Language) 形式に変換され、拡張子 .wsdl を持つようにリネームされます。ビジネス・オブジェクト定義ファイルは .xsd 形式で保管されます。

  これらのファイルを WebSphere Studio Application Developer Integration Edition で使用するには、81 ページの『新しいサービス・プロジェクト用のファイルのインポート』で説明しているとおり、サービス・プロジェクトにインポートする必要があります。

3. いずれかのオプションを選択し、「完了 (Finish)」をクリックします。

   指定した場所にファイルが生成されます。生成されるファイルは、インターフェース用 .wsdl ファイル、JMS バインディング用 .wsdl ファイル、および JMS サービス用 .wsdl ファイルです。これらのファイルは、すべてコネクター構成に固有です。また、ビジネス・オブジェクト定義ごとに、対応する .xsd ファイルが生成されます。

WebSphere Application Server アプリケーションの作成

このタスクでは、対話パターンを実装する WebSphere Application Server アプリケーションを作成します。このアプリケーションには、以下のタイプのプロジェクトが含まれます。

- サービス・プロジェクト

  System Manager から配置された初期 .wsdl ファイルと初期 XSD ファイルのほか、別途生成される、EJB バインディングが定義された .wsdl ファイルを含みます。別途生成された helper クラスを含む場合もあります。

- Enterprise Application Archive プロジェクト

1 つ以上の EJB プロジェクトで構成されます (この後生成する Session Bean と Message-Driven Bean (MDB) を含みます)。

WebSphere Application Server コンポーネント (EJB と MDB) のほかにも、以下の helper クラスがツールによって生成されます。

- wsdl ファイルまたは xsd ファイル内で検出された XML スキーマ定義ごとに生成され、検出されたスキーマによってのみ決まるフォーマットを持つデータ型 Java Bean。
- データ型 Bean とワイヤー・フォーマットの間でシリアライズとデシリアライズを実行するために使用される、フォーマット・ハンドラー Java Bean。ワイヤー・フォーマットは、エンコード方式とバインディングに応じて異なります (.wsdl ファイルに指定されている内容に従います)。したがって、フォーマット・ハンドラーも、エンコード方式とバインディングに応じて異なります。ここで説明するソリューションでは、フォーマット・ハンドラー・クラスのパッケージ名が示すように、バインディングは JMS、エンコード方式は XML です。

プロジェクトの作成が完了すると、個々の対話パターンを Enterprise Application Archive アプリケーションとしてエクスポートし、WebSphere Application Server に配置することができます。複数の対話パターンを単一の Enterprise Application Archive アプリケーションに組み込んで配置することもできます。

注: 最終的な Enterprise Application Archive プロジェクトには、ヘルパー・ファイル boutils.jar が含まれていなければなりません。このファイルには、ビジネス・オブジェクト初期化ライブラリー・ルーチンが含まれています。このファイルは、本章の『ビジネス・オブジェクト初期化ライブラリーの追加』の説明に従い、完成した Enterprise Application Archive に手動インポートで追加する必要があります。

このセクションの各トピックでは、対話パターンごとに作成方法を説明します。

エージェント・デリバリー (非同期イベント・デリバリー)

エージェント・デリバリー対話パターンでは、アダプターから WebSphere Application Server へメッセージが非同期配信されます。WebSphere Application Server からの応答メッセージは必要とされません。

エージェント・デリバリー対話パターンを作成するには、以下のものを含む Enterprise Application Archive アプリケーションを作成する必要があります。

- メッセージを受け取って EJB を呼び出す Message-Driven Bean (MDB)
- イベント処理に使用する必要があるビジネス・ロジックを実装しており、MDB によって呼び出される EJB
- .xsd ファイルを Java Bean として表現する helper クラス、および .wsdl ファイルに指定されている内容に従って XML を Java に変換するフォーマット・ハンドラー Java Bean

作成する Enterprise Application Archive アプリケーションには、さらに、サービス・プロジェクトから .wsdl ファイルと .xsd ファイルを組み込まなければなりません。System Manager のユーザー・プロジェクトを配置するときに「サービス・プロジェクトへエクスポート」を選択していれば、サービス・プロジェクトはすでに存

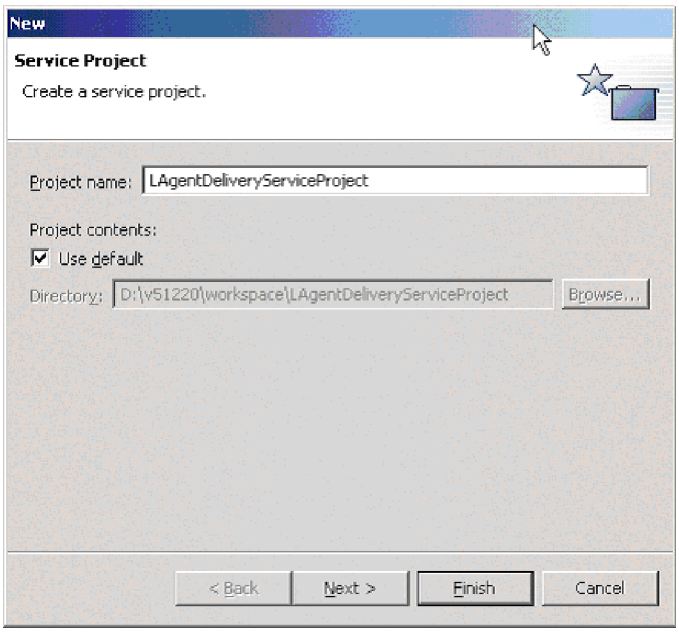
在し、Enterprise Application Archive に組み込まれています。このようにせず、System Manager のユーザー・プロジェクトを個々のファイルに分けてディレクトリーにエクスポートした場合や、JAR ファイルとしてエクスポートした場合には、それらの個々のファイルまたは JAR ファイルを新しいサービス・プロジェクトにインポートする必要があります。

新しいサービス・プロジェクト用のファイルのインポート

System Manager のユーザー・プロジェクトを既存のサービス・プロジェクトに直接エクスポートした場合には (78 ページの『ユーザー・プロジェクトの配置』を参照)、このステップは不要です。System Manager から JAR ファイルまたはディレクトリーへエクスポートを行った場合には、このステップを実行して、サービス・プロジェクトにファイルをインポートする必要があります。このステップでは、Session Bean と、この Bean について記述した .wsdl ファイルを作成します。

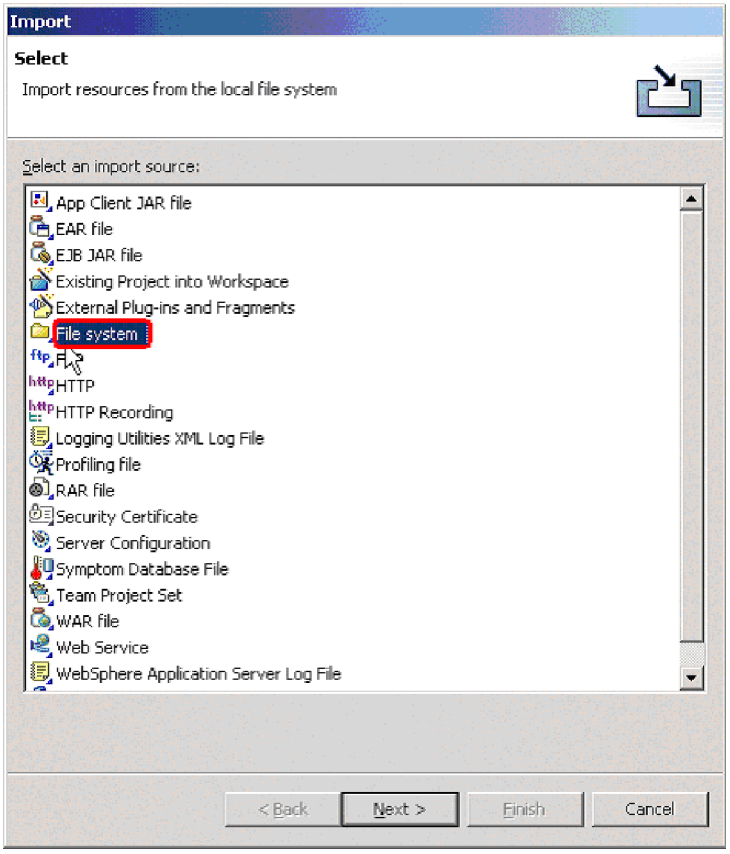
以下は、System Manager で「ディレクトリーへエクスポート」を選択した場合の手順です。エクスポートされたファイルを新しいサービス・プロジェクトにインポートするには、以下の操作を行います。

1. Business Integration パースペクティブで、「ファイル」>「新規」>「サービス・プロジェクト (Service Project)」を選択します。
2. 「新規 (サービス・プロジェクト (Service Project))」ダイアログが表示されます。プロジェクト名として使用する名前を入力します。「プロジェクトの内容 (Project contents)」の設定はデフォルトのままにして、「次へ」をクリックします。



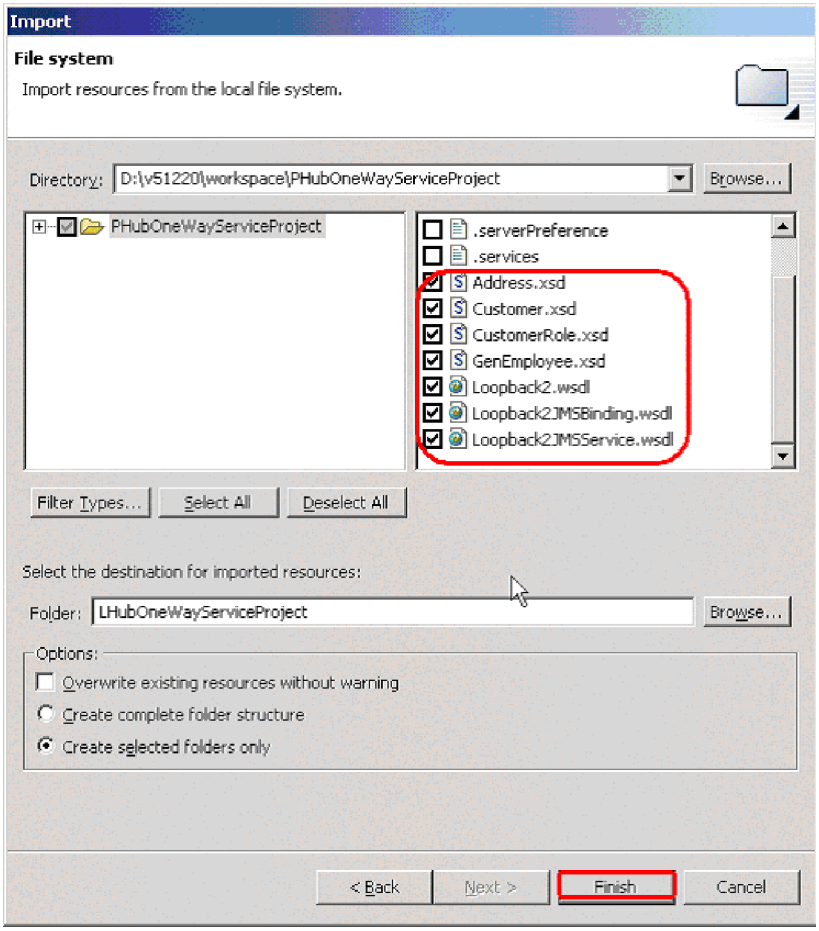
3. 「完了 (Finish)」をクリックします。新しいサービス・プロジェクトがデフォルト・ワークスペースに作成されます。Business Integration パースペクティブのパッケージ・エクスプローラー・ビューに、中核となるライブラリーが表示されます。既存のプロジェクトから組み込んだ追加の要素があれば、それも表示されます。

4. パッケージ・エクスプローラー・ビューで、作成した新しいサービス・プロジェクトを右マウス・ボタンでクリックし、「インポート」を選択します。
5. 「インポート」ダイアログが表示されます。「ファイル・システム」フォルダーを選択し、「次へ」をクリックします。



6. 「ディレクトリー」スクロール・ボックスで、目的の対話パターンで使用するために作成した .wsdl ファイルと .xsd ファイルが格納されているフォルダーを参照によって探し出します。System Manager からユーザー・プロジェクトを配置する方法で、目的の対話パターン用の .xsd ファイルと .wsdl ファイルを作成した場合には、そのユーザー・プロジェクトと同じ名前のフォルダーがすでに存在しています。そのフォルダーを選択し、「OK」をクリックします。
7. 「インポート」ダイアログに、選択したプロジェクト・フォルダーに含まれるファイルが表示されます。チェック・ボックスを使用して、個々の .xsd ファイルまたは .wsdl ファイルごとに組み込むか除外するかを指定することができます。デフォルトでは、System Manager で作成されたユーザー・プロジェクトには現在作成中の対話パターンで使用できる要素のみが組み込まれていると想定されるため、フォルダー内に表示されたすべての .xsd ファイルと .wsdl ファイルが選択されます。ただし、このダイアログでは、任意のファイルのみを選択することも、すべてのファイルを選択することもできます。通常、サポートされるビジネス・オブジェクトとしてコネクター構成内に指定したビジネス・オブジェクトの .xsd ファイルは、すべて、そのコネクターの構成ファイルから作成された WSDL ファイルと同じフォルダーに格納されています。サポートされているビジネス・オブジェクトの .xsd ファイルのいずれかが WSDL ファイルの格納先フォルダーから欠落している場合には、エラー・メッセージが表示

されることがあります。



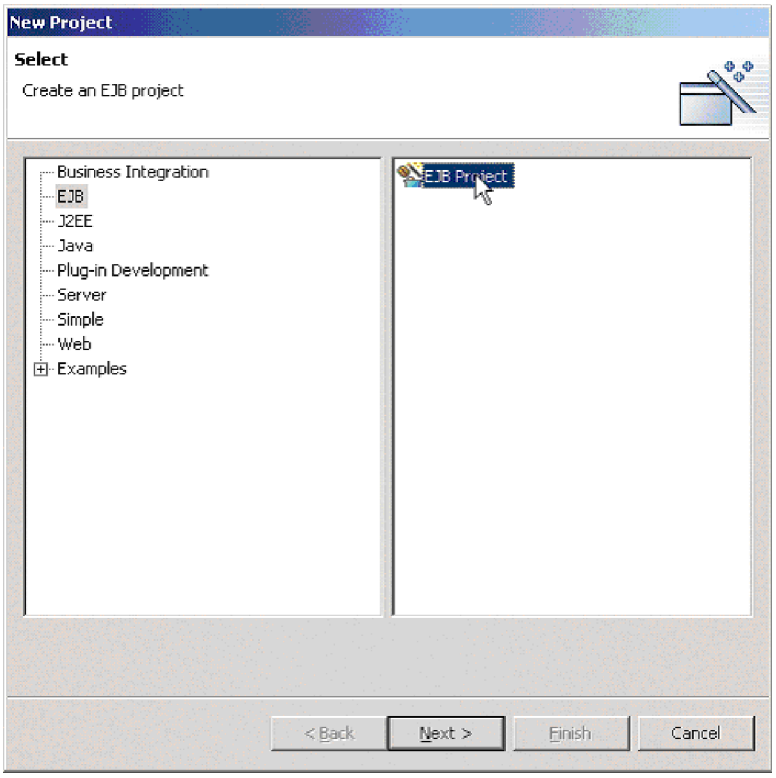
8. インポートするファイルの宛先フォルダーの名前を指定します。例えば、MyServices¥com¥ibm¥cw と入力します。
9. 「完了 (Finish)」をクリックします。
10. パッケージ・エクスプローラー内の指定したサービス・プロジェクトの下に、インポートしたファイルが表示されます。

EJB の作成

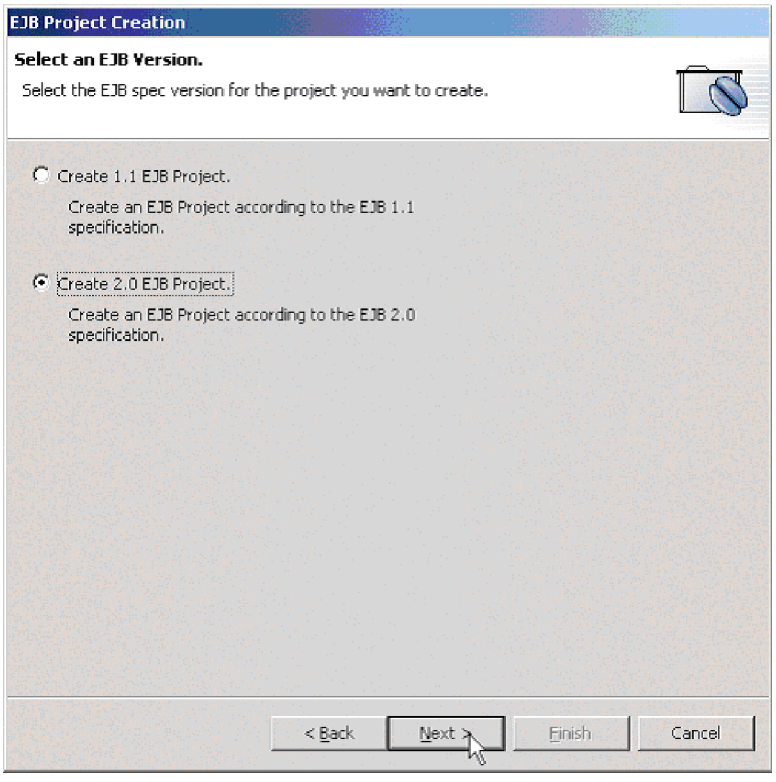
エージェント・デリバリー対話パターンでは、エージェントから WebSphere Application Server に着信するメッセージを消費するために EJB が必要です。EJB を作成する作業と、EJB の受信イベント処理メソッドにビジネス・ロジックを追加する作業の両方を行う必要があります。

まず、以下の手順で、EJB が格納されるプロジェクトを作成します。

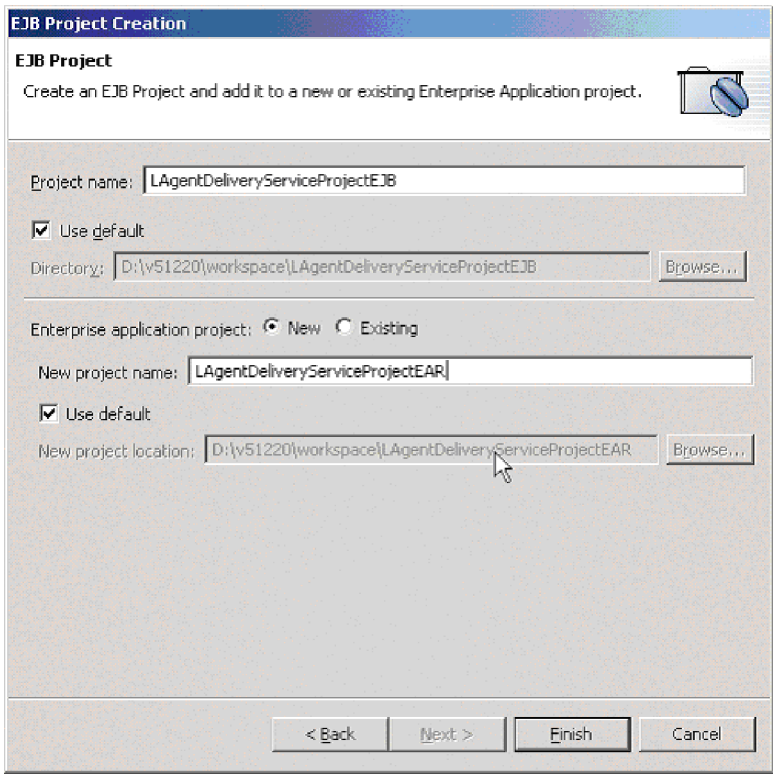
1. Business Integration パースペクティブの J2EE ビューで、「ファイル」>「新規」>「プロジェクト」を選択します。
2. 「新規プロジェクト」ダイアログで、左側のパネルの「EJB」を選択してから右側のパネルの「EJB プロジェクト (EJB project)」を選択し、「次へ」を選択します。



3. 「EJB バージョンの選択 (Select an EJB Version)」ダイアログで、「2.0 EJB プロジェクトを作成 (Create 2.0 EJB Project)」を選択し、「次へ」をクリックします。

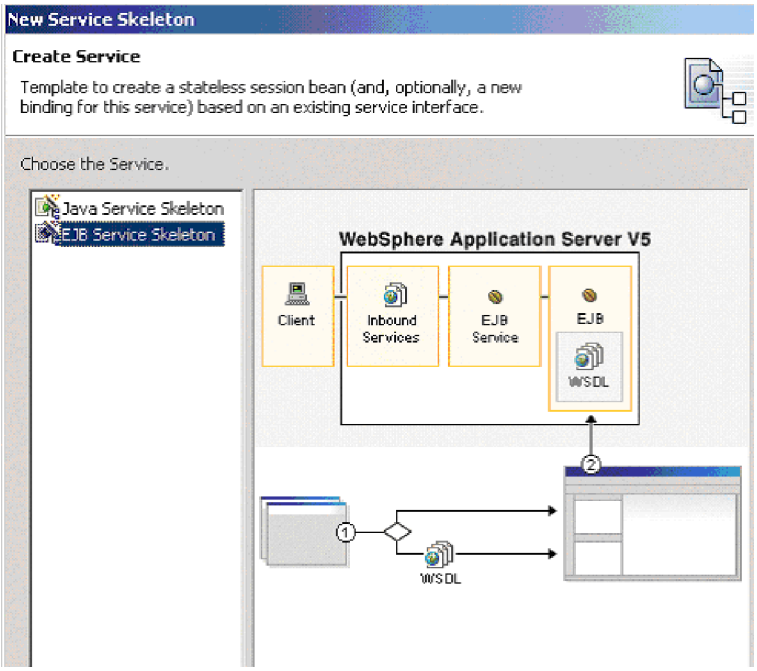


4. 「EJB プロジェクトの作成 (EJB Project Creation)」ダイアログで、作成する EJB プロジェクトの名前と、その EJB プロジェクトの追加先になる新規または既存の Enterprise Application Archive プロジェクト・フォルダーの名前を指定します。新規 Enterprise Application Archive プロジェクト名を指定すると、そのプロジェクト名を持つフォルダーが作成されます。



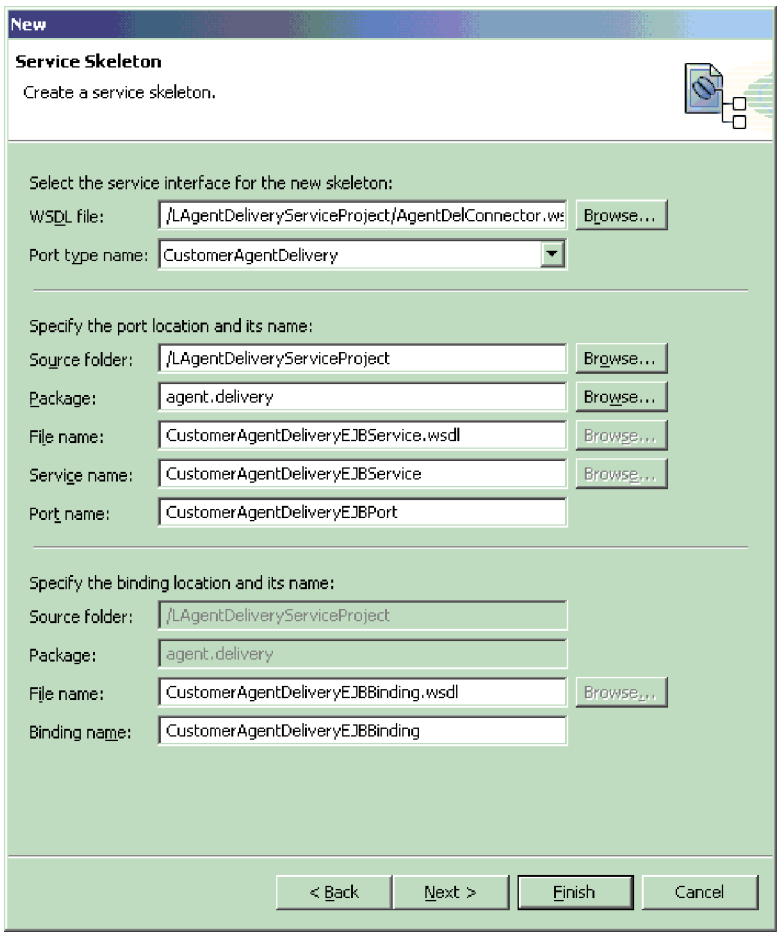
5. 「完了 (Finish)」を選択します。EJB モジュールが含まれるプロジェクトが、指定した名前のプロジェクト・フォルダーの下に生成されます。作成された EJB プロジェクトは、J2EE 階層ビューの「EJB モジュール (EJB Modules)」の下に表示されます。サービス・ビューの「サービス・プロジェクト (Service Projects)」の下で、コネクター構成からインポートされた .wsdl インターフェース・ファイル (AgentDelConnector.wsdl など) を選択します。右マウス・ボタンでクリックし、「新規」>「サービスからビルド (Build from Service)」を選択します。「新規サービス・スケルトン (New Service Skeleton)」ダイアログが開きます。

6. 「EJB サービス・スケルトン (EJB Service Skeleton)」を選択し、「次へ」をクリックします。



7. 「新規サービス・スケルトン (New Service Skeleton)」ダイアログで、「*新規ポートおよびバインディングを作成 (Create a new port and binding)*」および「*helper クラスを生成 (Generate helper classes)*」を選択します。「次へ」を選択します。

8. 「サービス・スケルトン (Service Skeleton)」ダイアログが表示されます。



9. 作成しようとしている対話パターンに適したポート・タイプ名を選択し、デフォルトのパッケージ名の代わりに分かりやすいパッケージ名を指定します。「次へ」をクリックします。
10. 次のダイアログでは、EJB スケルトンの生成に関するプロパティーを指定できます。デフォルト値をそのまま使用することもできます。「完了 (Finish)」をクリックします。新しく作成された EJB プロジェクトが、サービス・ビューの「配置可能なサービス (Deployable Services)」の下のリストに表示されます。
11. スケルトンを編集して、必要なビジネス・ロジックを追加します。

MDB の作成

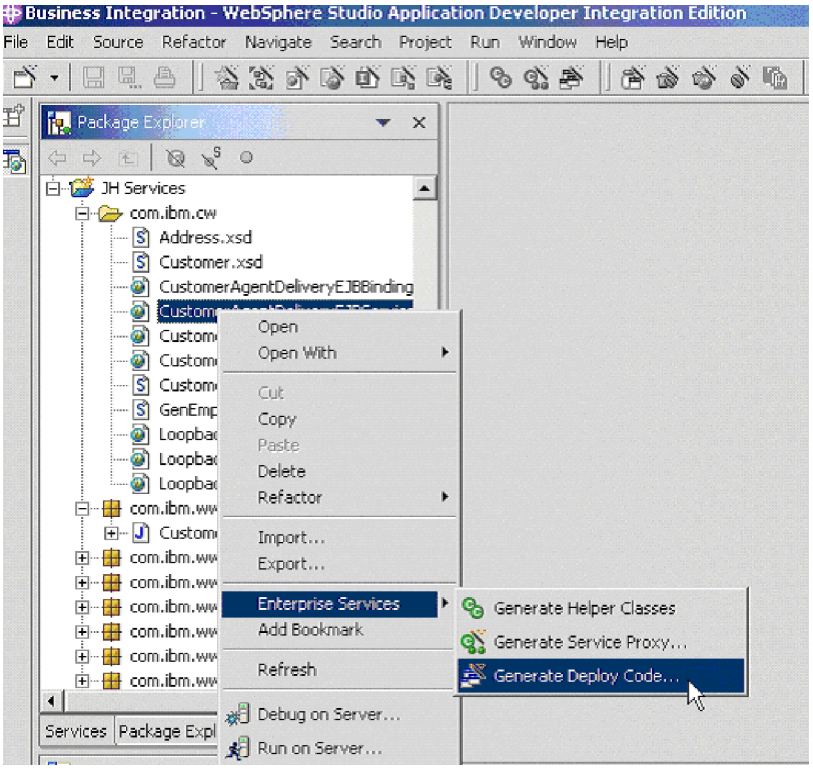
Message-Driven Bean は、コネクターからイベントをメッセージとして受け取り、EJB を呼び出してそれらのイベントを処理させる役割を果たします。

このステップでは、既存のインバウンド・ポートとバインディング (オリジナルの JMS バインディング) を使用して、EJB サービスをサーバーに配置します。このステップを実行すると、Session Bean スケルトンの適切なメソッドを呼び出す MDB が生成されます。生成される MDB は、ポート・タイプに応じて異なります。また、操作は、着信メッセージのプロパティーである WSDLOperation プロパティーに基づいて選択されます (WSDLInput プロパティーと WSDLOutput プロパティーも関係することがあります)。そのほかに、helper クラスやフォーマット・ハンドラーも配置の際に生成されます。これらは、ワイヤー・フォーマット (XML メッセー

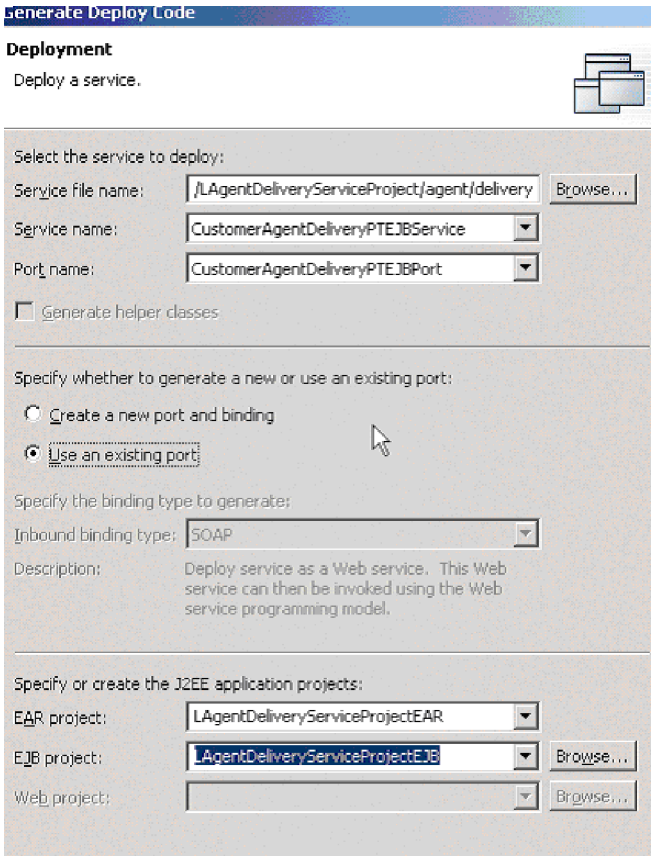
ジ) から Java クラスへの変換のために使用されます。変換後、Java クラスは EJB のビジネス・メソッドに引き数として渡されます。

EJB と MDB で使用される JNDI 名を構成する必要があります。この JNDI のセットアップが必要なのは、WSDL で定義されているキュー接続ファクトリーとキューのルックアップ名が、サーバーに定義されている実際の名前と一致していないことがあるためです。WebSphere Application Server では、ルックアップする名前 (WSDL で定義されている名前) を、サーバー上で配置されている実際の JNDI 名にマップできます。したがって、wsdl ファイルに変更を加えなくても、正確にルックアップを実行できます。また、MDB が使用するリスナー・ポートも指定する必要があります。

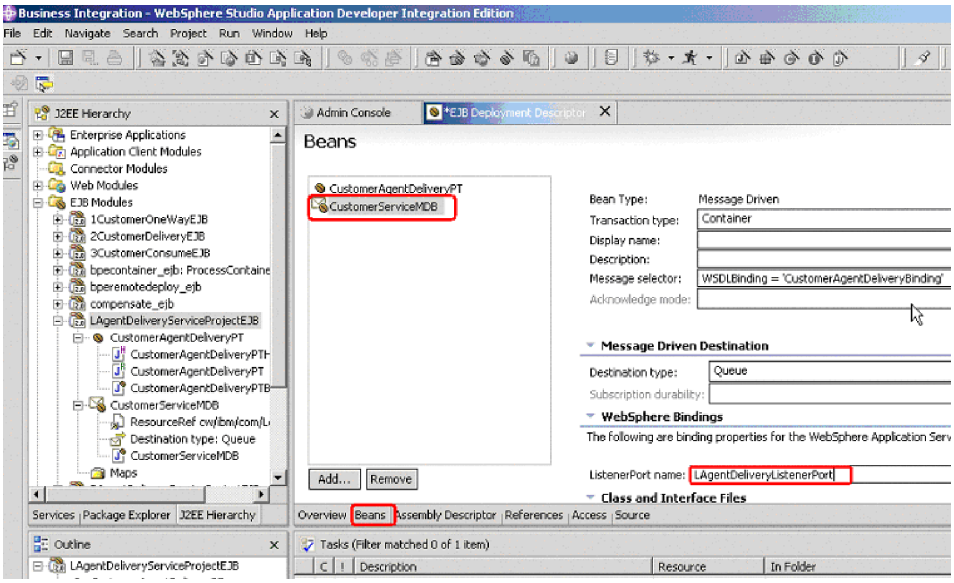
1. パッケージ・エクスプローラー・ビューで、作成したサービス・プロジェクトのフォルダーを展開し、直前のタスクで生成された EJB サービスの .wsdl ファイルを選択します。選択したファイルを右マウス・ボタンでクリックし、ポップアップで「エンタープライズ・サービス (Enterprise Services)」>「配置コードの生成 (Generate Deploy Code)」を選択します。



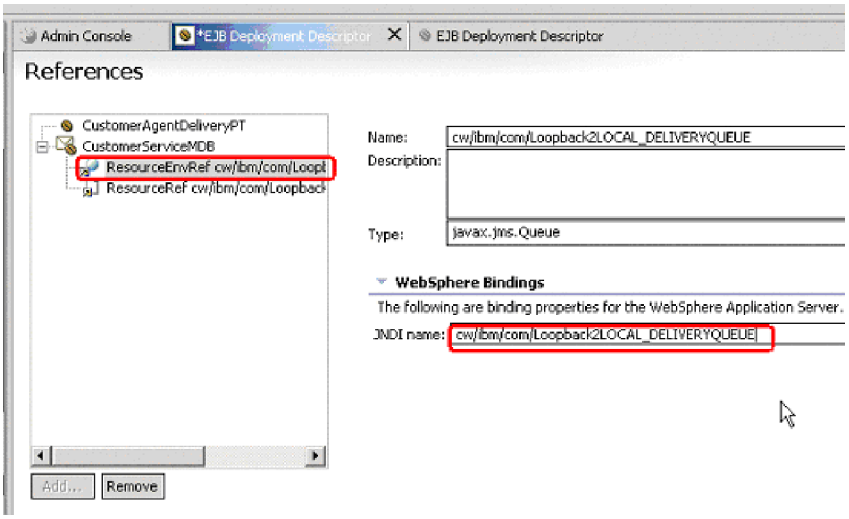
2. 「配置コードの生成 (Generate Deploy Code)」ダイアログで、「*既存のポートを使用する (Use an existing port)*」を選択し、「次へ」を選択します。



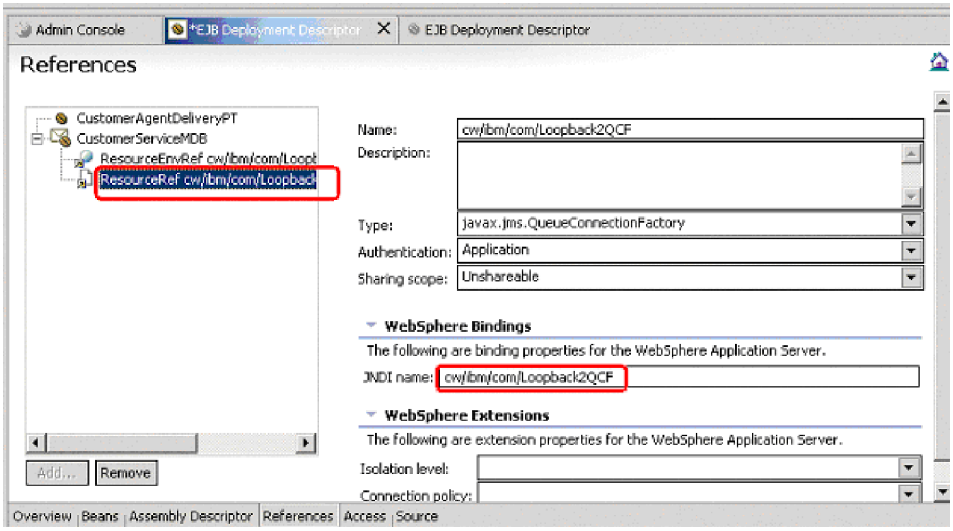
3. 「インバウンド・サービス・ファイル (Inbound Service Files)」ダイアログで、現在のプロジェクト用の JMS サービスの .wsdl ファイルを選択し、対応するサービス名とポート名を選択します。「完了 (Finish)」を選択します。
4. J2EE 階層ビューの「EJB モジュール (EJB Modules)」の下でプロジェクトを選択し、右マウス・ボタンでクリックします。「*EJB 配置記述子エディターで開く (Open With the EJB Deployment Descriptor editor)*」を選択し、「Bean」タブを選択します。「宛先タイプ (Destination type)」に「キュー (Queue)」と表示されていることを確認します。適切なリスナー・ポートを指定します。リスナー・ポートのキュー接続ファクトリーおよびキューが、目的の対話パターン用の .wsdl ファイルに指定されているキュー接続ファクトリーおよびキューと一致するようにします。



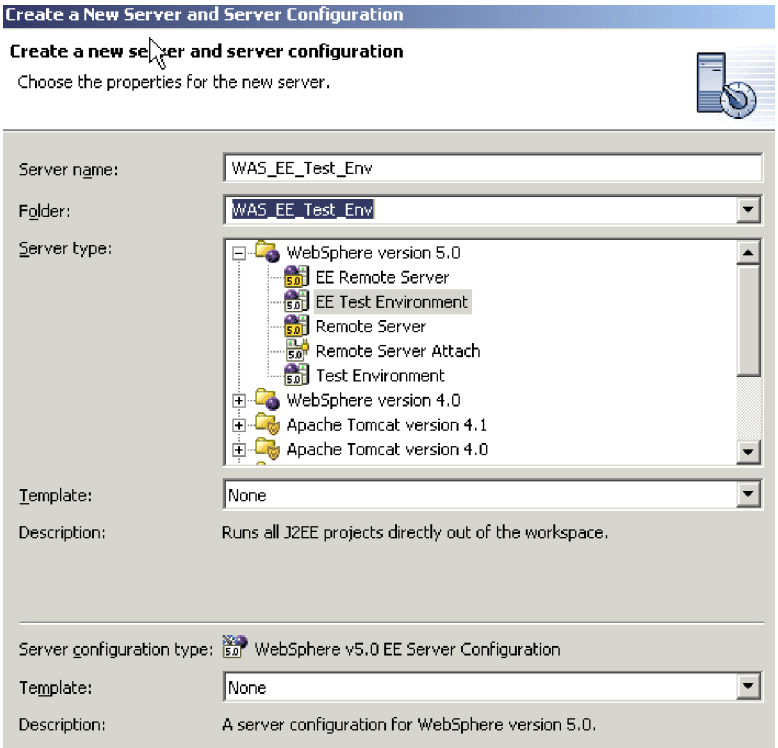
5. 「参照」タブを選択します。MDB の下に表示されたリストから、リソース・キュー名を選択します。.wsdl ファイルから取得された名前が、右側の「*名前*」ボックスに表示されます。「WebSphere バインディング (WebSphere Bindings)」の下の「*JNDI name (JNDI 名)*」フィールドに、選択したキューを配置するための JNDI 名を、WebSphere Application Server 管理コンソールで定義する内容と一致するように入力します。



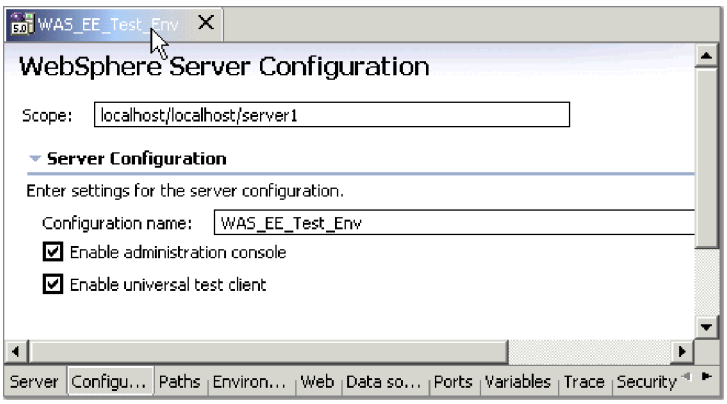
6. MDB の下に表示されたリストから、接続ファクトリーを選択します。「WebSphere バインディング (WebSphere Bindings)」の下の「*JNDI name (JNDI 名)*」フィールドに、この接続ファクトリーを配置するための JNDI 名を、WebSphere Application Server 管理コンソールで定義した内容と一致するように入力します。



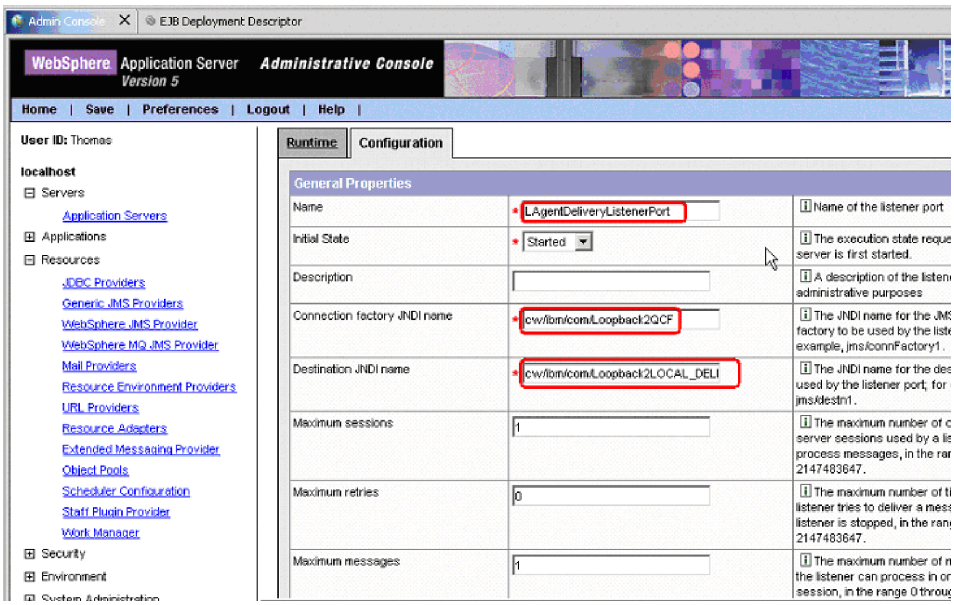
7. EJB 配置記述子エディターで行った変更を保管します。
8. Business Integration パースペクティブの下側のパネルにある「サーバー構成 (Server Configuration)」タブを開き、「サーバー」アイコンを右マウス・ボタンでクリックします。
9. 「新規」を選択し、「サーバーとサーバー構成 (Servers and Server Configuration)」を選択します。「*新規のサーバーとサーバー構成を作成 (Create a New Server and Server Configuration)*」ダイアログが表示されます。



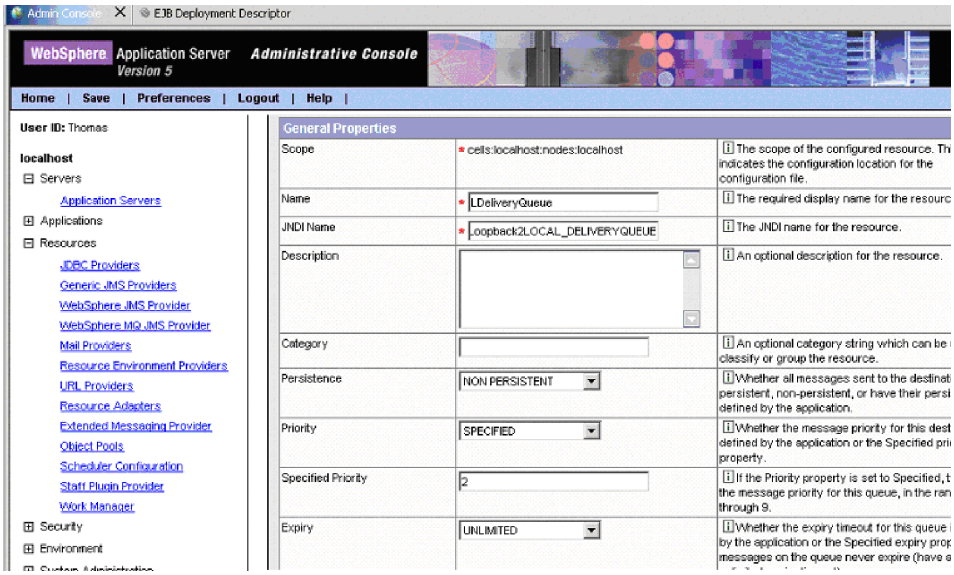
10. 「サーバー名」フィールドで、作成するサーバーのインスタンス名を入力します。同じ名前を「フォルダー」フィールドにも入力します。「サーバー・タイプ (Server Type)」フィールドで、「EE テスト環境 (EE Test Environment)」を選択します。
11. 「完了 (Finish)」を選択し、指定した名前を付けて新規のサーバー・プロジェクトを作成するようにプロンプトが出されたら、「はい」と応答します。
12. ユーティリティーが完了したら、「サーバー構成 (Server Configuration)」パネルで、作成した新規のサーバー・インスタンスのアイコンをダブルクリックします。「WebSphere サーバー (WebSphere Server)」パネルと、作成したサーバー・インスタンスの名前が表示されます。「構成」タブを選択します。
13. 「WebSphere サーバー構成 (WebSphere Server Configuration)」パネルが表示されます。「管理コンソールを使用可能にする (Enable administration console)」チェック・ボックスにチェックマークを付けます。



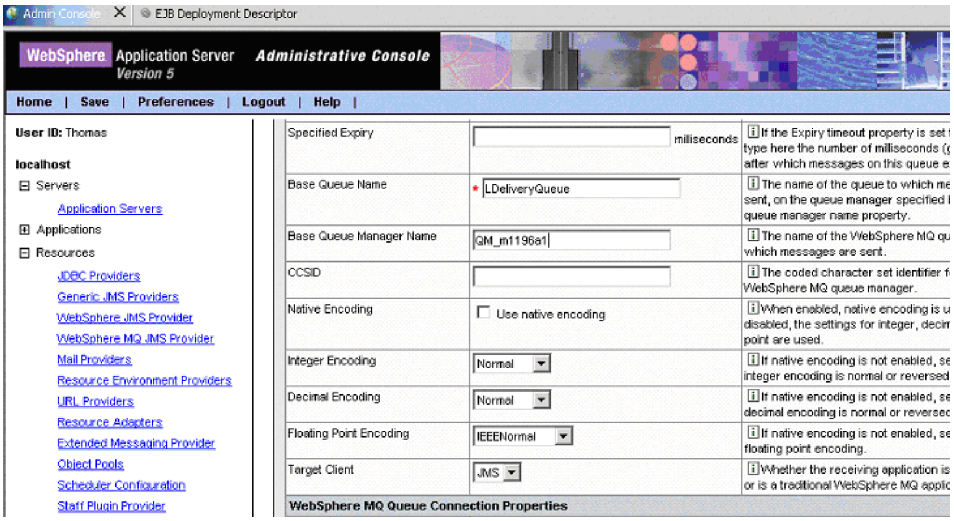
14. 「変数」タブを選択します。「定義済みの変数 (Defined variables)」ドロップダウン・ボックスで、MQ_INSTALL_ROOT 変数を探します。変数の値を MQ インストール・フォルダーまでの絶対ルート・パスに変更します。
15. 構成を保管します。(Ctrl キーを押しながら S を押します)。
16. 新規に構成したサーバー・インスタンスを次のようにして始動します。Business Integration パースペクティブの下部で、「サーバー」タブを選択します。「サーバー」パネルが表示されます。「サーバー」パネルで、新規に構成したサーバー・インスタンスのアイコンを選択し、「実行」ボタンをクリックしてサーバーを始動します。
17. WebSphere Application Server 管理コンソールを開きます。「構成」タブの「一般プロパティー (General Properties)」に、EJB 配置記述子エディターの「Bean」タブで指定したリスナー・ポート名を追加します。また、EJB 配置記述子エディターの「参照」タブで指定した接続ファクトリーの JNDI 値および宛先キューの JNDI 名値と同じ値を追加します。必要に応じて、「最大セッション数 (Maximum sessions)」、「最大再試行数 (Maximum retries)」、「最大メッセージ数 (Maximum messages)」の各フィールドで設定の変更を行います。



18. WebSphere Application Server 管理コンソールで、キューをリソースとして追加します。キューの表示名と JNDI ルックアップ名を入力します。この JNDI ルックアップ名は、EJB 配置記述子とリスナー・ポートに指定した名前と同じにする必要があります。



「基本キュー名 (Base Queue Name)」フィールドに、MQ キュー・マネージャーに作成した物理キューの名前を入力します。このキューは、Connector Configurator を使用してコネクター構成プロパティーに指定した DeliveryQueue と一致する必要があります。



「JNDI 名 (JNDI Name)」フィールドに、EJB 配置記述子エディターの「Bean」タブで宛先キューの JNDI 名値として設定したキュー名を入力します。「基本キュー・マネージャー名 (Base Queue Manager Name)」の値を指定します。

19. EJB 配置記述子とリスナー・ポートに使用した名前に対応する JNDI 名を付けて、キュー接続ファクトリーを作成します。

ここまでのステップを完了すると、目的の対話パターンに対応する Enterprise Application Archive アプリケーションが完成します。この Enterprise Application Archive アプリケーションは、その後、EE テスト環境でテストしたり、別の WebSphere Application Server 環境に Enterprise Application Archive ファイルとして配置したりできます。

エージェント要求 (同期イベント・デリバリー)

エージェント要求対話パターンでは、アダプターから WebSphere Application Server に要求メッセージが同期配信されます。このとき、応答が要求されます。

エージェント要求対話パターンを作成するには、以下のものを含む Enterprise Application Archive アプリケーションを作成する必要があります。

- メッセージを受け取って EJB を呼び出す Message-Driven Bean (MDB)
- イベント処理に使用する必要があるビジネス・ロジックを実装しており、MDB によって呼び出される EJB
- .xsd ファイルを Java Bean として表現する helper クラス、および .wsdl ファイルに指定されている内容に従って XML を Java に変換するフォーマット・ハンドラー Java Bean

作成する Enterprise Application Archive アプリケーションには、さらに、サービス・プロジェクトから .wsdl ファイルと .xsd ファイルを組み込まなければなりません。System Manager のユーザー・プロジェクトを配置するときに「サービス・プロジェクトへエクスポート」を選択していれば、サービス・プロジェクトはすでに存在し、Enterprise Application Archive に組み込まれています。このようにせず、

System Manager のユーザー・プロジェクトを個々のファイルに分けてディレクトリーにエクスポートした場合や、JAR ファイルとしてエクスポートした場合には、それらの個々のファイルまたは JAR ファイルを新しいサービス・プロジェクトにインポートする必要があります。これを行うには、エージェント・デリバリー対話パターン用に説明したインポート手順と同じ手順を実行してください (81 ページの『新しいサービス・プロジェクト用のファイルのインポート』を参照)。

エージェント要求対話パターンの作成手順は、エージェント・デリバリー対話パターンの作成手順と同様です。ただし、エージェント要求対話パターンでは、応答メッセージの送信が必要です。したがって、エージェント要求対話パターン用の EJB スケルトンに含まれるビジネス・ロジックは、値を戻さなければなりません。

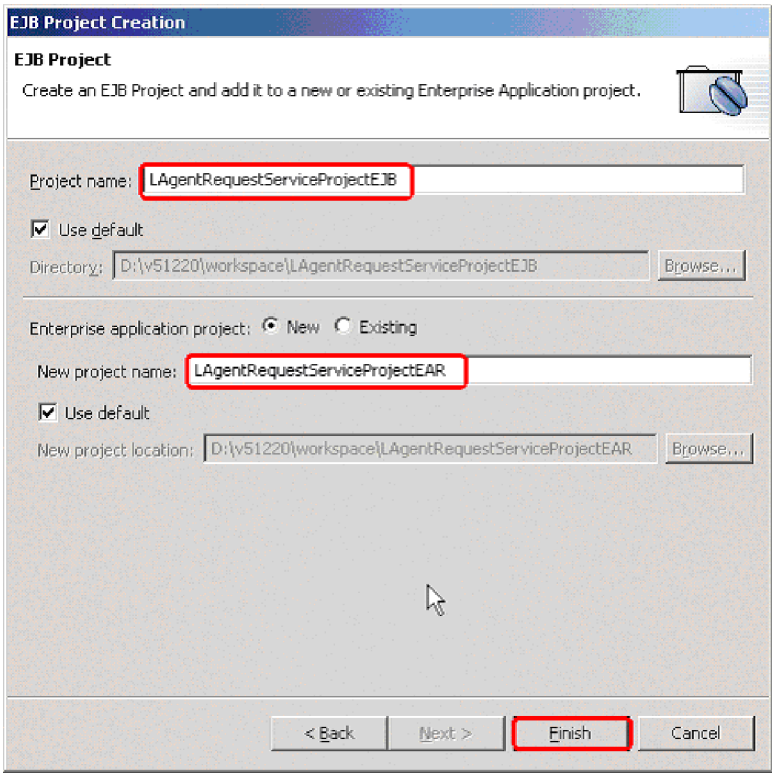
エージェント要求対話パターンを作成するには、以下の手順を実行します。

EJB プロジェクトの作成

エージェント要求対話パターンでは、エージェントから WebSphere Application Server に着信するメッセージを消費するために EJB が必要です。EJB を作成する作業と、EJB の受信イベント処理メソッドにビジネス・ロジックを追加する作業の両方を行う必要があります。

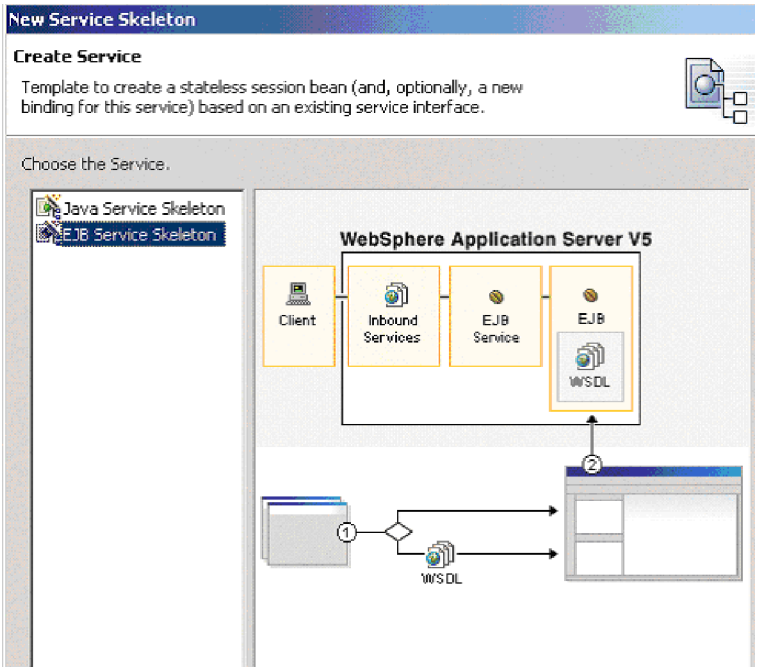
まず、以下の手順で、EJB が格納されるプロジェクトを作成します。

1. Business Integration パースペクティブの J2EE ビューで、「ファイル」>「新規」>「プロジェクト」を選択します。
2. 「新規プロジェクト」ダイアログで、左側のパネルの「EJB」を選択してから右側のパネルの「EJB プロジェクト (EJB project)」を選択し、「次へ」を選択します。
3. 「EJB バージョンの選択 (Select an EJB Version)」ダイアログで、「2.0 EJB プロジェクトを作成 (Create 2.0 EJB Project)」を選択し、「次へ」をクリックします。
4. 「EJB プロジェクトの作成 (EJB Project Creation)」ダイアログで、作成する EJB プロジェクトの名前と、その EJB プロジェクトの追加先になる新規または既存の Enterprise Application Archive プロジェクト・フォルダーの名前を指定します。新規 Enterprise Application Archive プロジェクト名を指定すると、そのプロジェクト名を持つフォルダーが作成されます。



5. 「完了 (Finish)」を選択します。EJB モジュールが含まれるプロジェクトが、指定した名前のプロジェクト・フォルダーの下に生成されます。作成された EJB プロジェクトは、J2EE 階層ビューの「EJB モジュール (EJB Modules)」の下に表示されます。サービス・ビューの「サービス・プロジェクト (Service Projects)」の下で、コネクター構成からインポートされた .wsdl インターフェース・ファイル (AgentReqConnector.wsdl など) を選択します。右マウス・ボタンでクリックし、「新規」>「サービスからビルド (Build from Service)」を選択します。「新規サービス・スケルトン (New Service Skeleton)」ダイアログが開きます。

6. 「EJB サービス・スケルトン (EJB Service Skeleton)」を選択し、「次へ」をクリックします。



7. 「新規サービス・スケルトン (New Service Skeleton)」ダイアログで、「*新規ポートおよびバインディングを作成 (Create a new port and binding)*」および「*helper クラスを生成 (Generate helper classes)*」を選択します。「次へ」を選択します。
8. 「サービス・スケルトン (Service Skeleton)」ダイアログが表示されます。
9. 作成しようとしている対話パターンに適したポート・タイプ名を選択し、デフォルトのパッケージ名の代わりに分かりやすいパッケージ名を指定します。「次へ」をクリックします。
10. 次のダイアログでは、EJB スケルトンの生成に関するプロパティーを指定できます。デフォルト値をそのまま使用することもできます。「完了 (Finish)」をクリックします。新しく作成された EJB プロジェクトが、サービス・ビューの「配置可能なサービス (Deployable Services)」の下のリストに表示されます。
11. スケルトンを編集して、必要なビジネス・ロジックを追加します。

MDB の作成

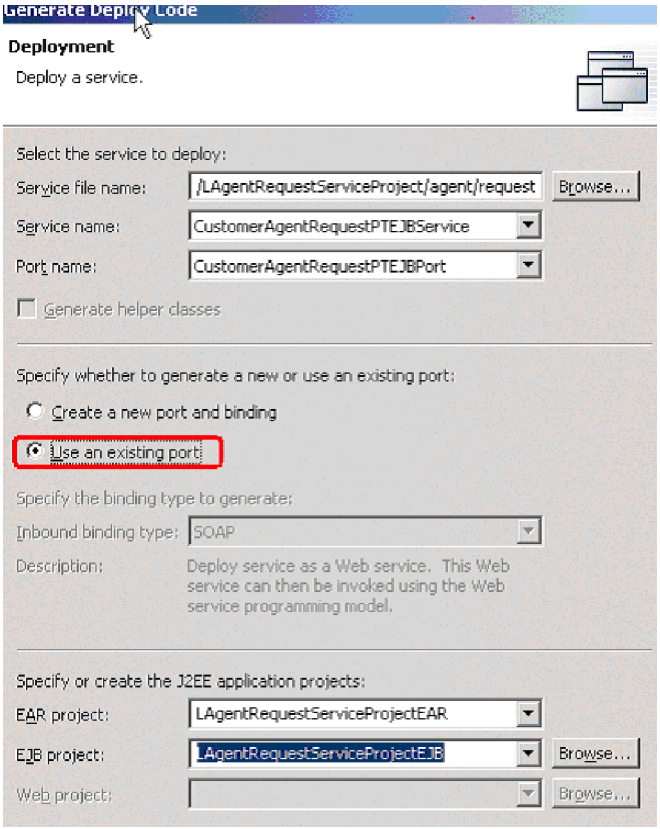
Message-Driven Bean は、コネクターからイベントをメッセージとして受け取り、EJB を呼び出してそれらのイベントを処理させる役割を果たします。

このステップでは、既存のインバウンド・ポートとバインディング (オリジナルの JMS バインディング) を使用して、EJB サービスをサーバーに配置します。このステップを実行すると、Session Bean スケルトンの適切なメソッドを呼び出す MDB が生成されます。生成される MDB は、ポート・タイプに応じて異なります。また、操作は、着信メッセージのプロパティーである WSDLOperation プロパティーに基づいて選択されます (WSDLInput プロパティーと WSDLOutput プロパティーも関係することがあります)。そのほかに、helper クラスやフォーマット・ハンドラーも配置の際に生成されます。これらは、ワイヤー・フォーマット (XML メッセー

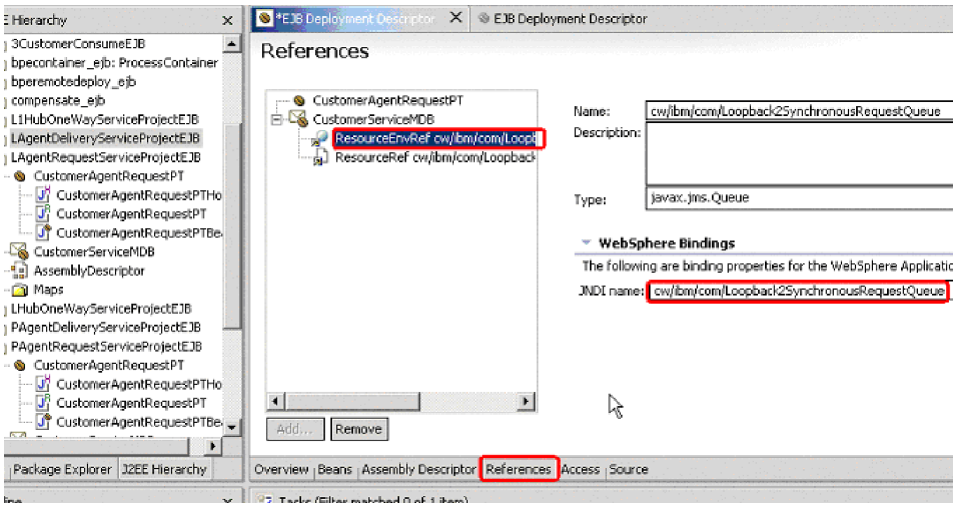
ジ) から Java クラスへの変換のために使用されます。変換後、Java クラスは EJB のビジネス・メソッドに引き数として渡されます。

EJB と MDB で使用される JNDI 名を構成する必要があります。この JNDI のセットアップが必要なのは、WSDL で定義されているキュー接続ファクトリーとキューのルックアップ名が、サーバーに定義されている実際の名前と一致していないことがあるためです。WebSphere Application Server では、ルックアップする名前 (.wsdl ファイル内に定義されている名前) を、サーバー上で採用されている実際の JNDI 名にマップできます。したがって、wsdl ファイルに変更を加えなくても、正確にルックアップを実行できます。また、MDB が使用するリスナー・ポートも指定する必要があります。

1. パッケージ・エクスプローラー・ビューで、自分が作成したサービス・プロジェクトのフォルダーを展開し、直前のタスクで生成された EJB サービスの .wsdl ファイルを選択します。選択したファイルを右マウス・ボタンでクリックし、ポップアップで「エンタープライズ・サービス (Enterprise Services)」>「配置コードの生成 (Generate Deploy Code)」を選択します。
2. 「配置コードの生成 (Generate Deploy Code)」ダイアログで、「*既存のポートを使用する (Use an existing port)*」を選択します。直前のステップで EJB スケルトン用に作成した EJB プロジェクトと同じプロジェクトを選択することも、名前を指定して新規 EJB プロジェクトを作成することもできます。新規 EJB プロジェクトを作成する場合、そのプロジェクトは、既存の EJB スケルトン用プロジェクトと同じ Enterprise Application Archive プロジェクトになければなりません。「次へ」を選択します。

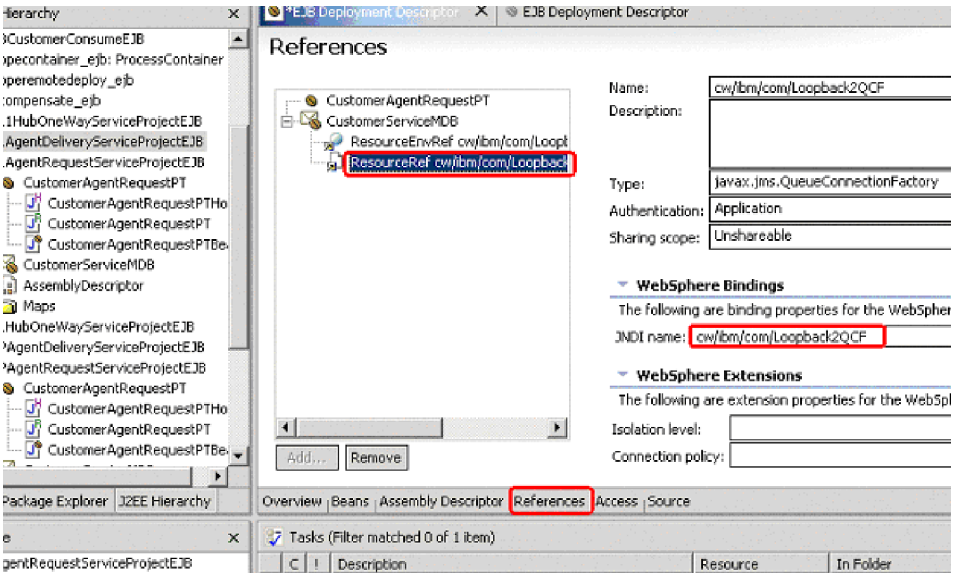


3. 「インバウンド・サービス・ファイル (Inbound Service Files)」ダイアログで、現在のプロジェクト用の JMS サービスの .wsdl ファイルを選択し、対応するサービス名とポート名を選択します。「完了 (Finish)」を選択します。
4. J2EE 階層ビューの「EJB モジュール (EJB Modules)」の下でプロジェクトを選択し、右マウス・ボタンでクリックします。「*EJB 配置記述子エディターで開く (Open With the EJB Deployment Descriptor editor)*」を選択し、「Bean」タブを選択します。「宛先タイプ (Destination type)」に「キュー (Queue)」と表示されていることを確認します。適切なリスナー・ポートを指定します。リスナー・ポートのキュー接続ファクトリーおよびキューが、目的の対話パターン用の .wsdl ファイルに指定されているキュー接続ファクトリーおよびキューと一致するようにします。
5. 「参照」タブを選択します。MDB の下に表示されたリストから、リソース・キュー名を選択します。.wsdl ファイルから取得された名前が、右側の「*名前*」ボックスに表示されます。「WebSphere バインディング (WebSphere Bindings)」の下の「*JNDI name (JNDI 名)*」フィールドに、選択したキューを配置するための JNDI 名を、WebSphere Application Server 管理コンソールで定義する内容と一致するように入力します。



6. MDB の下に表示されたリストから、接続ファクトリーを選択します。「WebSphere バインディング (WebSphere Bindings)」の下の「*JNDI name (JNDI 名)*」フィールドに、この接続ファクトリーを配置するための JNDI 名を、WebSphere Application Server 管理コンソールで定義した内容と一致するよ

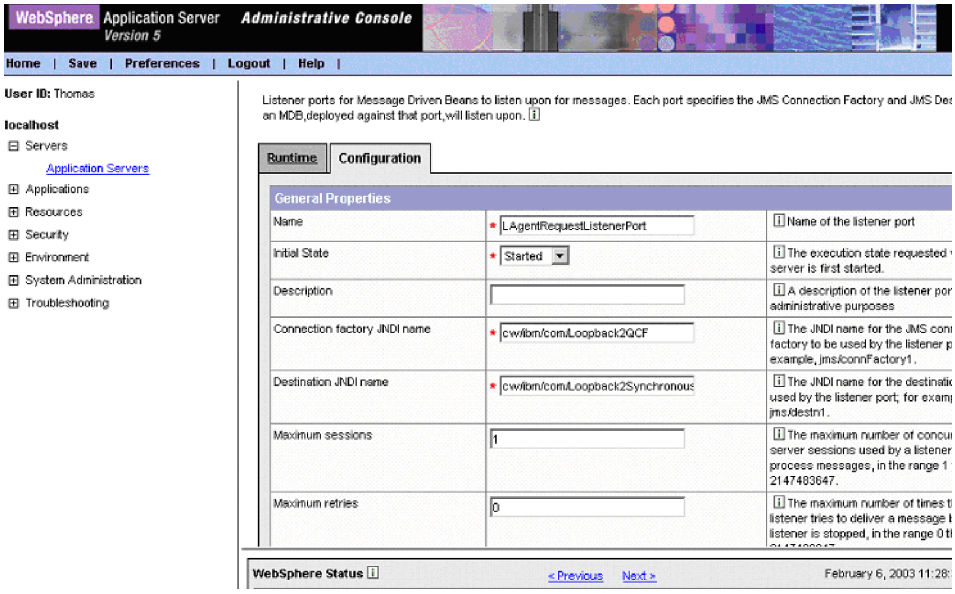
うに入力します。



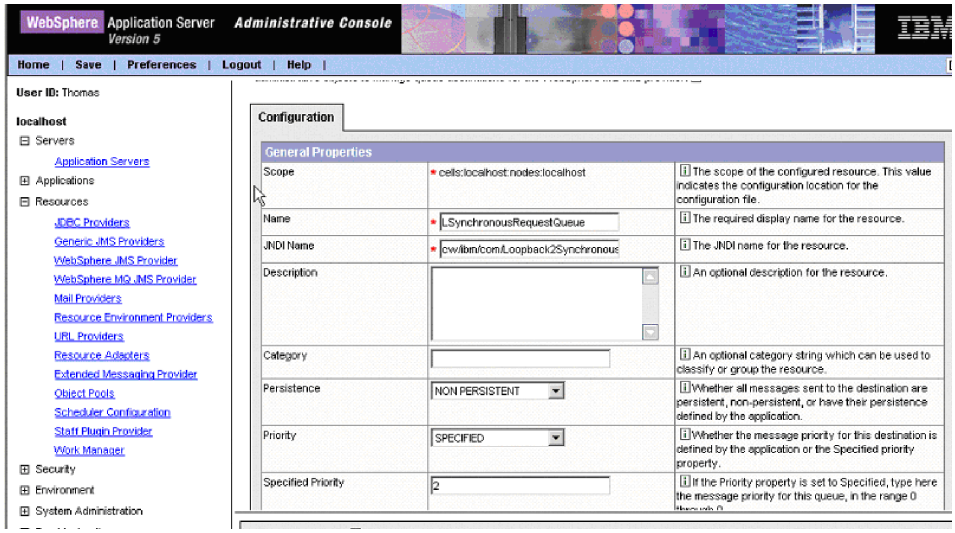
7. EJB 配置記述子エディターで行った変更を保管します。
8. Business Integration パースペクティブの下側のパネルにある「サーバー構成 (Server Configuration)」タブを開き、「サーバー」アイコンを右マウス・ボタンでクリックします。
9. 「新規」を選択し、「サーバーとサーバー構成 (Servers and Server Configuration)」を選択します。*「新規のサーバーとサーバー構成を作成 (Create a New Server and Server Configuration)」*ダイアログが表示されます。
10. 「サーバー名」フィールドで、作成するサーバーのインスタンス名を入力します。同じ名前を「フォルダー」フィールドにも入力します。「サーバー・タイプ (Server Type)」フィールドで、「EE テスト環境 (EE Test Environment)」を選択します。
11. 「完了 (Finish)」を選択し、指定した名前を付けて新規のサーバー・プロジェクトを作成するようにプロンプトが出されたら、「はい」と応答します。
12. ユーティリティーが完了したら、「サーバー構成 (Server Configuration)」パネルで、作成した新規のサーバー・インスタンスのアイコンをダブルクリックします。「WebSphere サーバー (WebSphere Server)」パネルと、作成したサーバー・インスタンスの名前が表示されます。「構成」タブを選択します。
13. 「WebSphere サーバー構成 (WebSphere Server Configuration)」パネルが表示されます。「管理コンソールを使用可能にする (Enable administration console)」チェック・ボックスにチェックマークを付けます。
14. 「変数」タブを選択します。「定義済みの変数 (Defined variables)」ドロップダウン・ボックスで、MQ_INSTALL_ROOT 変数を探します。変数の値を MQ インストール・フォルダーまでの絶対ルート・パスに変更します。
15. 構成を保管します。(Ctrl キーを押しながら S を押します)。
16. 新規に構成したサーバー・インスタンスを次のようにして始動します。Business Integration パースペクティブの下部で、「サーバー」タブを選択します。「サ

ーバー」パネルが表示されます。「サーバー」パネルで、新規に構成したサーバー・インスタンスのアイコンを選択し、「実行」ボタンをクリックしてサーバーを始動します。

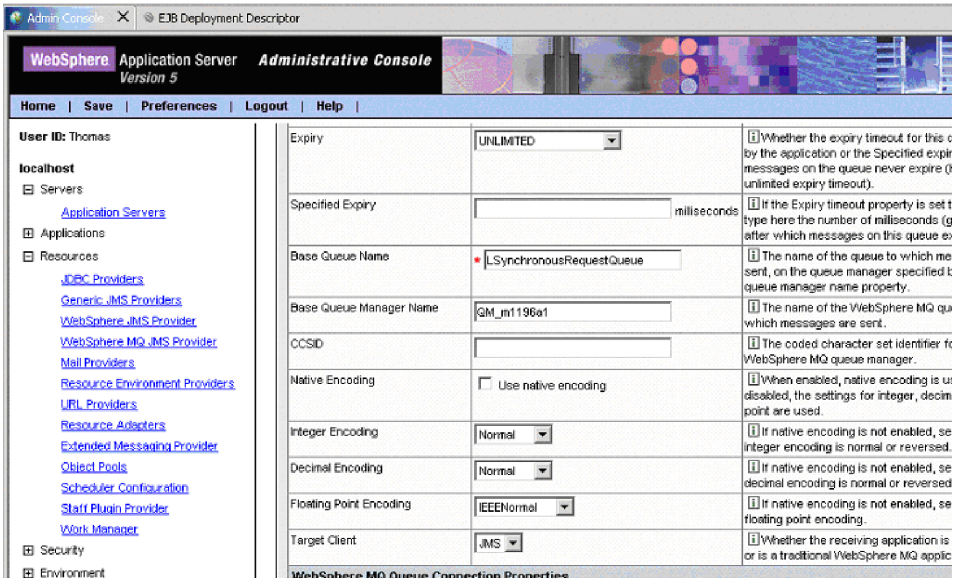
17. WebSphere Application Server 管理コンソールを開きます。「構成」タブの「一般プロパティー (General Properties)」に、EJB 配置記述子エディターの「Bean」タブで指定したリスナー・ポート名を追加します。また、EJB 配置記述子エディターの「参照」タブで指定した接続ファクトリーの JNDI 値および宛先キューの JNDI 名値と同じ値を追加します。必要に応じて、「最大セッション数 (Maximum sessions)」、「最大再試行数 (Maximum retries)」、「最大メッセージ数 (Maximum messages)」の各フィールドで設定の変更を行います。



18. WebSphere Application Server 管理コンソールで、キューをリソースとして追加します。キューの表示名を指定し、それと同じ値を「基本キュー名 (Base Queue Name)」フィールドに入力します。「JNDI 名 (JNDI Name)」フィールドに、EJB 配置記述子エディターの「Bean」タブで宛先キューの JNDI 名値として設定したキュー名を入力します。



「基本キュー・マネージャー名 (Base Queue Manager Name)」の値を指定します。



ここまでのステップを完了すると、目的の対話パターンに対応する Enterprise Application Archive アプリケーションが完成します。その後、EE テスト環境でこの Enterprise Application Archive アプリケーションをテストしたり、別の WebSphere Application Server 環境に Enterprise Application Archive アプリケーションを配置したりできます。

ハブ片方向

ハブ片方向対話パターンでは、WebSphere Application Server からアダプターにメッセージが配信されますが、アダプターからの応答は必要とされません。

この対話パターン用の Enterprise Application Archive アプリケーションには、サービス・プロジェクトが含まれていなければなりません。System Manager のユーザー・プロジェクトを配置するときに「サービス・プロジェクトへエクスポート」を選択していれば、サービス・プロジェクトはすでに存在しています。System Manager のユーザー・プロジェクトを JAR ファイルとしてエクスポートした場合、またはファイルのセットとしてディレクトリーへエクスポートした場合には、そのファイルをインポートしてサービス・プロジェクトを作成する必要があります。この作業を行う必要がある場合には、エージェント・デリバリー対話パターン用にこの作業の手順を説明している、81 ページの『新しいサービス・プロジェクト用のファイルのインポート』を参照してください。これ以降のハブ片方向対話パターンを対象とした説明は、System Manager のユーザー・プロジェクトの配置時に「サービス・プロジェクトへエクスポート」が選択されていることを前提としています。

ハブ片方向対話パターンを作成するには、以下のものを含む Enterprise Application Archive アプリケーションを作成する必要があります。

- 任意の J2EE アプリケーションから使用できる、JMS サービスの呼び出しを行う Session Bean。この Session Bean は、配置される (JMS) ポート・タイプの操作ごとにビジネス・メソッドを備えていなければなりません。

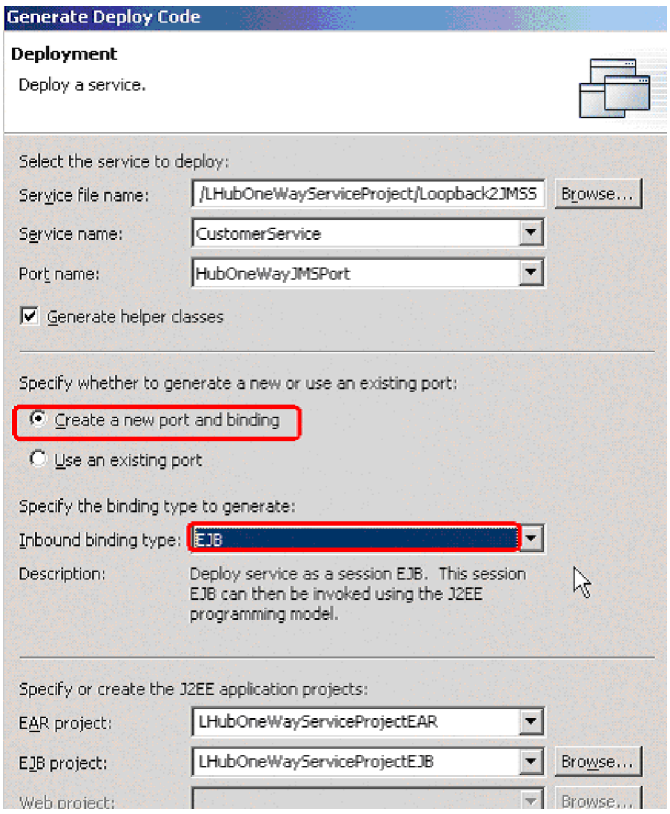
  Session Bean の作成が完了したら、.wsdl ファイルから取得されたキュー名および接続ファクトリー名と JNDI 名との間に必要になる WebSphere マッピングを行うため、配置記述子エディターを使用してその Session Bean を構成する必要があります。
- .xsd ファイルを Java Bean として表現する helper クラス、および .wsdl ファイルに指定されている内容に従って XML を Java に変換するフォーマット・ハンドラー Java Bean

上記の各作業については、以降の各トピックで説明します。

EJB の作成

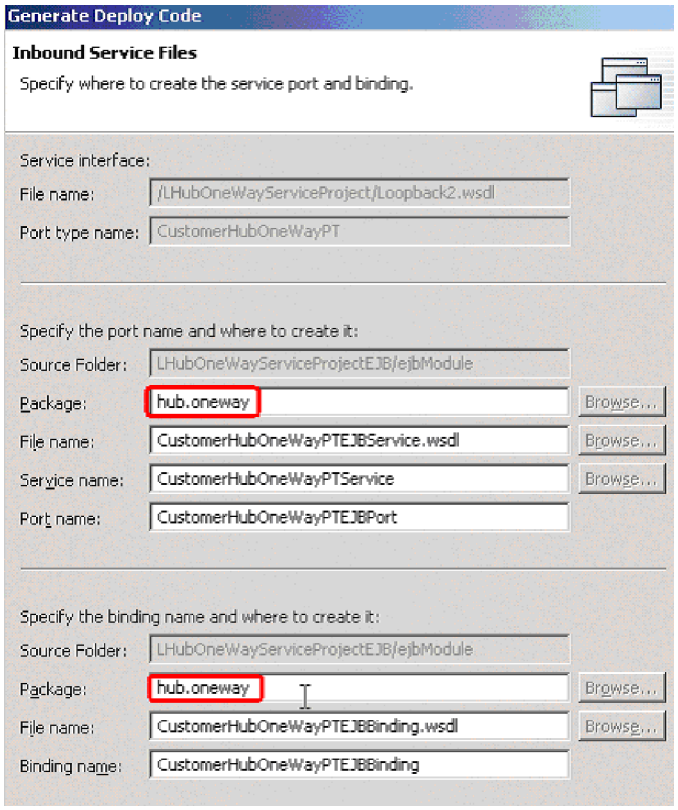
このステップでは、Session Bean と、この Bean について記述した .wsdl ファイル (EJB サービスおよびバインディング) を作成します。

1. パッケージ・エクスプローラー・ビューで、JMS サービスの .wsdl ファイルを選択します。(サービス・プロジェクトは、4 つの対話パターンのすべてで同じ内容です。そのため、ここで選択するファイルは、エージェント・デリバリー対話パターン用にサービス・プロジェクトを作成したときに生成された JMS サービス・ファイルと同じものにすることができます。)
2. 選択したファイルを右マウス・ボタンでクリックし、「エンタープライズ・サービス (Enterprise Services)」>「配置コードの生成 (Generate Deploy Code)」を選択します。「配置コードの生成 (Generate Deploy Code)」ダイアログが表示されます。



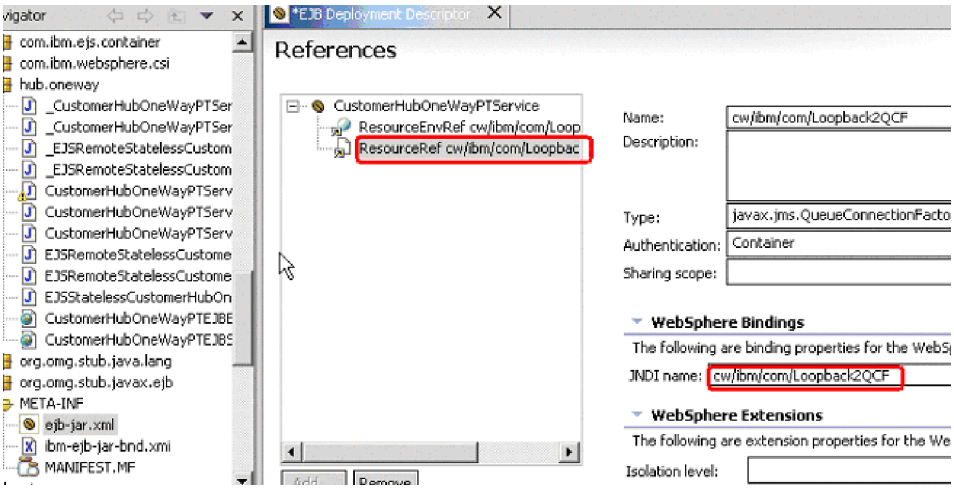
3. 「ポート名」ドロップダウンで、目的の対話パターンのポート名を選択します。
4. 「*新規ポートおよびバインディングを作成 (Create a new port and binding)*」を選択します。
5. 「*インバウンド・バインディング・タイプ (Inbound binding type)*」ドロップダウンで、「EJB」、「JMS」、または「SOAP」を選択します。選択した内容は、生成される WSDL ファイルに影響します (EJB アクセス用に EJB バインディングおよびサービスに関するファイルが生成されます)。

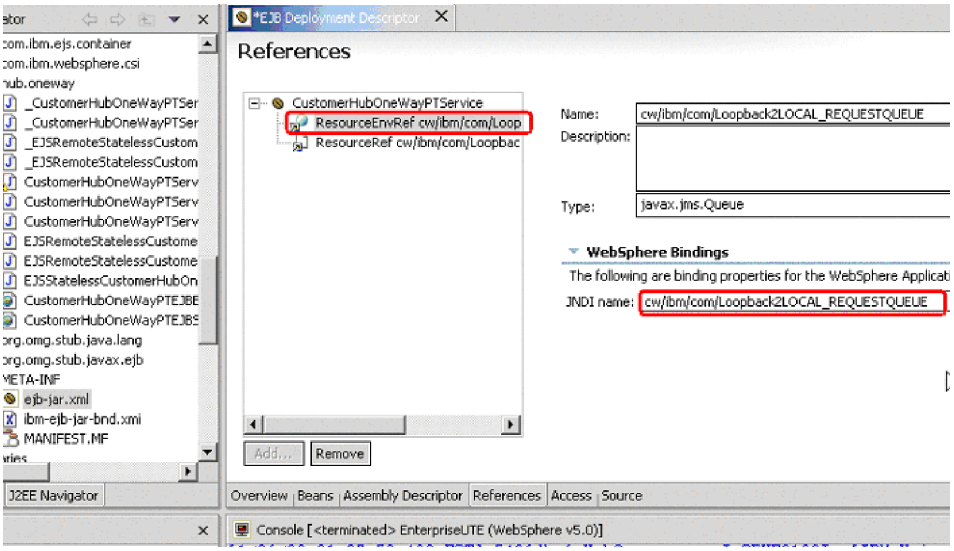
6. 「次へ」を選択します。「インバウンド・サービス・ファイル (Inbound Service Files)」ダイアログが表示されます。



7. 「インバウンド・サービス・ファイル (Inbound Service Files)」ダイアログで、ポート名とバインディング名の両方に、分かりやすいパッケージ名を指定します。
8. 「次へ」を選択します。「EJB ポート (EJB Port)」ダイアログが表示されます。このダイアログには、JNDI 名を EJB ポート・プロパティーの 1 つとして指定するためのフィールドがあります。デフォルト値をそのまま使用します。

9. 配置記述子エディターを使用して、キュー接続ファクトリー名およびキュー名に JNDI 名をマップします。





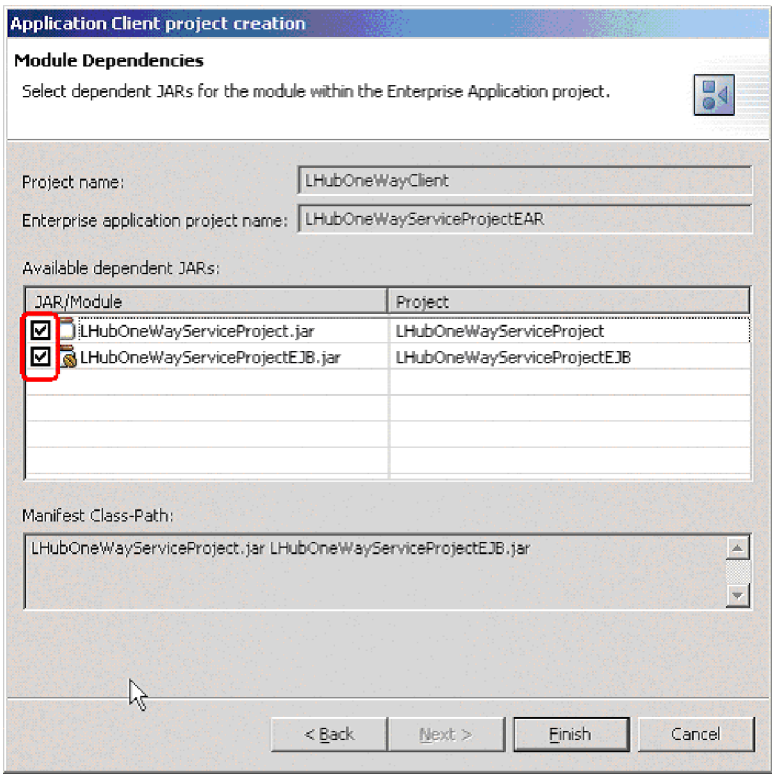
10. ハブ片方向を表現する EJB が完成しました。この EJB は、配置することも、別の WebSphere Application Server のコンポーネントとして使用することもできます。

テスト用アプリケーション・クライアント・プロジェクトの作成

完成したアプリケーション・パッケージをテストできるよう、クライアント・プロジェクトを Enterprise Application Archive ファイルに追加することができます (オプション)。Enterprise Application Archive クライアント・プロジェクトは、クライアント・クラスを使用して作成します。このクラスの main() メソッドのコードによって、EJB のルックアップや、EJB の適切なビジネス・メソッドの呼び出しが行われることになります。アプリケーション・クライアントを実行する場合は、クライアントの main() メソッドが含まれるクラスの名前を、アプリケーション・クライアントの配置記述子にあらかじめ指定しておく必要があります。

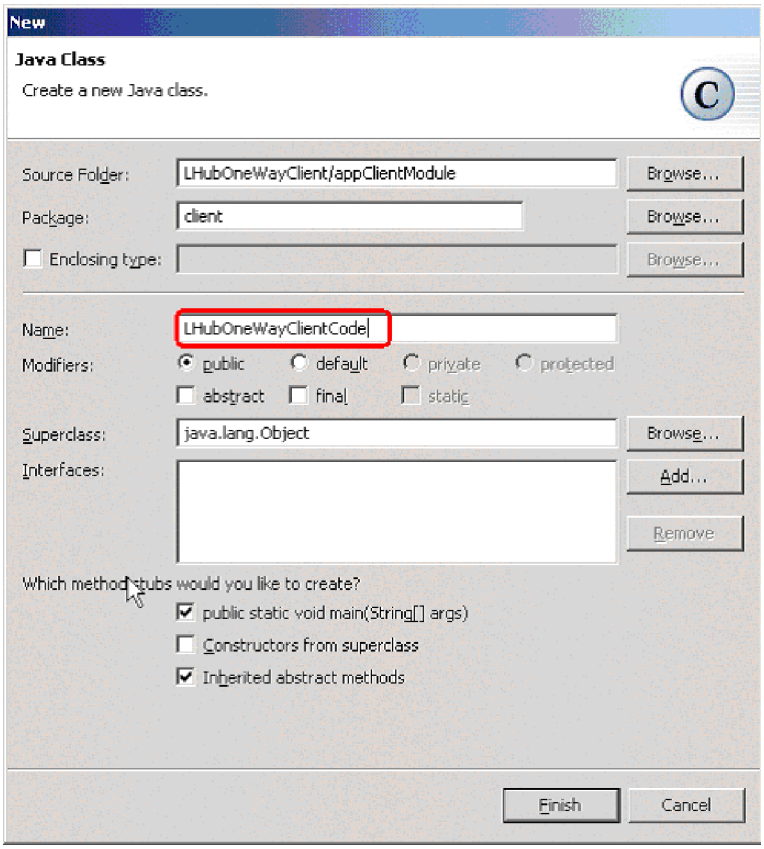
アプリケーション・クライアント・プロジェクトを作成するには、以下の手順を実行します。

1. パッケージ・エクスプローラー・ビューで、「ファイル」>「新規プロジェクト」を選択します。次に、「J2EE」>「アプリケーション・クライアント・プロジェクト (Application Client Project)」を選択します。「次へ」を選択します。
2. 「J2EE 仕様バージョン (J2EE Specification version)」ダイアログで、「J2EE 1.3 アプリケーション・クライアント・プロジェクトを作成 (Create J2EE 1.3 Application Client project)」を選択します。「次へ」を選択します。
3. 「アプリケーション・クライアント・プロジェクトの作成 (Application Client project creation)」ダイアログで、プロジェクト名を指定し、「既存 (Existing)」にチェックマークを付けます。直前のタスクで作成した EJB が含まれているソース Enterprise Application Archive プロジェクトの名前を参照によって探し出し、選択します。「次へ」を選択します。
4. 「モジュール依存関係 (Module Dependencies)」ダイアログが表示されます。このダイアログには、選択可能な依存 JAR ファイルのリストが表示されます。作成中のモジュールと共に Enterprise Application Archive プロジェクトに組み込む必要がある JAR ファイルのボックスにマークを付け、「完了 (Finish)」を選択します。

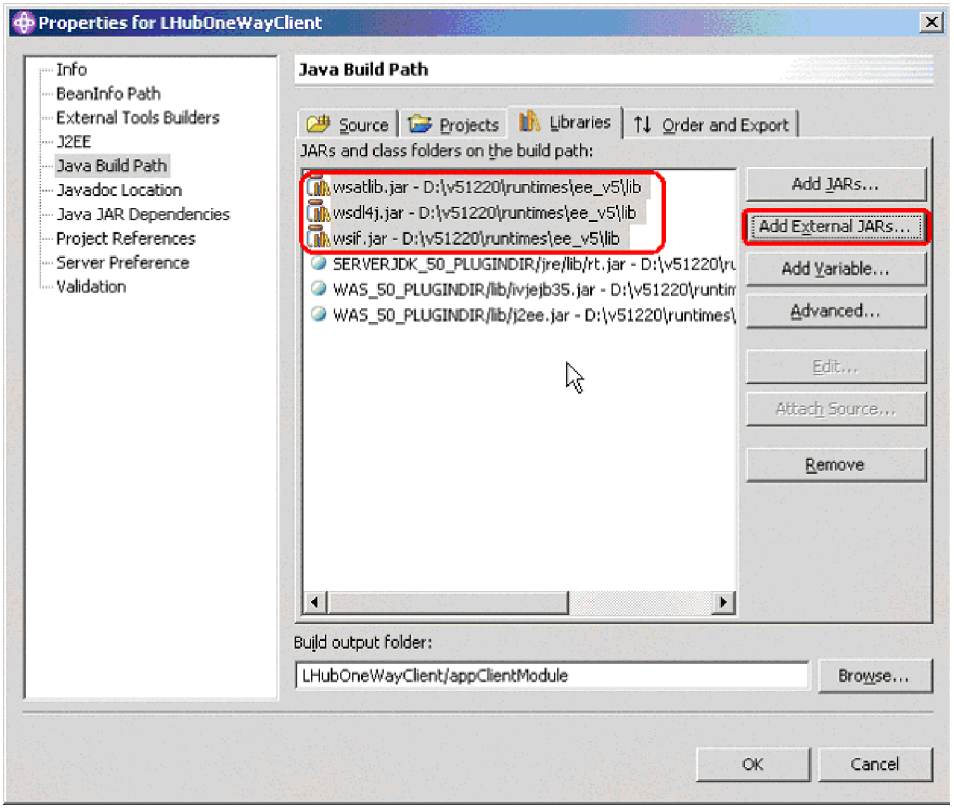


5. クライアント・アプリケーション内にパッケージを作成するには、パッケージ・エクスプローラー・ビューで、作成したアプリケーション・クライアント・モジュールを選択し、ツールバーの「Java パッケージの作成 (Create a Java Package)」ボタンを選択します。「新規 (Java パッケージ (New Java Package))」ダイアログが表示されます。パッケージ名を入力し、「完了 (Finish)」を選択します。入力したパッケージ名を持つフォルダーが、アプリケーション・クライアント・モジュールのフォルダーの下に追加されます。

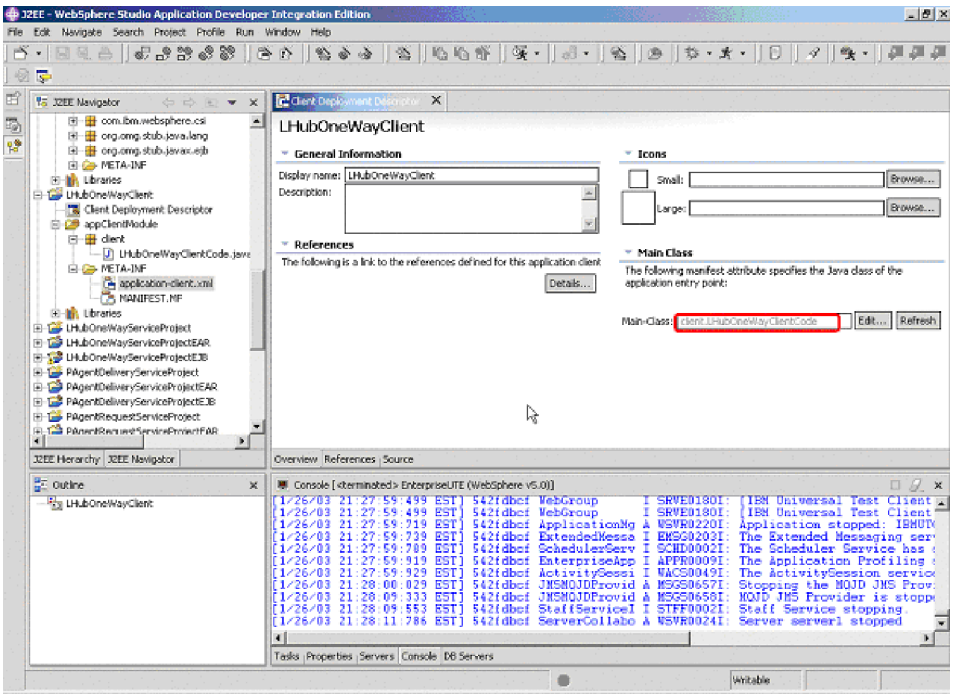
6. ロジックの追加先となる Java クラスを作成するには、アプリケーション・クライアントのフォルダーを選択し、メニュー・バーの「Java クラスの作成 (Create Java Class)」ボタンを選択します。
7. 「新規 (Java クラス (Java Class))」ダイアログが表示されます。「パッケージ」フィールドで、前述のステップで作成したクライアント・パッケージを選択します。クラス名を指定し、修飾子として「public」を指定します。さらに、作成するメソッド・スタブとして、「public static void」と「継承された抽象メソッド (inherited abstract methods)」を指定します。



8. 「完了 (Finish)」を選択します。
9. 作成した Java クラスを開き、ロジックを追加します。
10. 必要な JAR ファイルを追加します。これを行うには、パッケージ・エクスプローラーでアプリケーション・クライアントを選択して右マウス・ボタンでクリックし、「プロパティー」を選択して「Java ビルド・パス (Java Build Path)」を選択します。「Java ビルド・パス (Java Build Path)」パネルが表示されます。
11. 「ライブラリー (Libraries)」タブを選択し、「外部 JAR を追加 (Add External Jars)」を選択します。WebSphere Studio Application Developer Integration Edition をインストールしたディレクトリーに移動し、¥runtime¥ee_v5¥lib フォルダーを展開します。wsatlib.jar、wsdl4j.jar、wsif.jar、qname.jar などのファイルを選択します。



12. 「OK」をクリックします。
13. 配置記述子エディターを使用して、アプリケーション・クライアント・パッケージを更新します。
これを行うには、パッケージ・エクスプローラーで、アプリケーション・クライアント・モジュール内の META-INF フォルダーを展開し、application-client.xml ファイルを選択します。選択したファイルを右マウス・ボタンでクリックし、「アプリケーションから開く (Open With)」>「配置記述子エディター (Deployment Descriptor Editor)」を選択します。「クライアント配置記述子 (Client Deployment Descriptor)」ダイアログが表示されます。



14. メイン・クラス名を追加します。これで、アプリケーション・クライアントが使用可能になりました。

ハブ要求

ハブ要求対話パターンでは、WebSphere Application Server からアダプターへメッセージが同期配信され、応答の受信も行われます。

この対話パターン用の Enterprise Application Archive アプリケーションには、サービス・プロジェクトが含まれていなければなりません。System Manager のユーザー・プロジェクトを配置するときに「サービス・プロジェクトへエクスポート」を選択していれば、サービス・プロジェクトはすでに存在しています。System Manager のユーザー・プロジェクトを JAR ファイルとしてエクスポートした場合、またはファイルのセットとしてディレクトリーへエクスポートした場合には、そのファイルをインポートしてサービス・プロジェクトを作成する必要があります。この作業を行う必要がある場合には、エージェント・デリバリー対話パターン用にこの作業の手順を説明している、81 ページの『新しいサービス・プロジェクト用のファイルのインポート』を参照してください。これ以降のハブ要求対話パターンを対象とした説明は、System Manager のユーザー・プロジェクトの配置時に「サービス・プロジェクトへエクスポート」が選択されていることを前提としています。

ハブ要求対話パターンを作成するには、以下のものを含む Enterprise Application Archive アプリケーションを作成する必要があります。

- 任意の J2EE アプリケーションから使用できる、JMS サービスの呼び出しを行う Session Bean。この Session Bean は、配置される (JMS) ポート・タイプの操作ごとにビジネス・メソッドを備えていなければなりません。

Session Bean の作成が完了したら、.wsdl ファイルから取得されたキュー名および接続ファクトリー名と JNDI 名との間に必要になる WebSphere マッピングを行うため、配置記述子エディターを使用してその Session Bean を構成する必要があります。

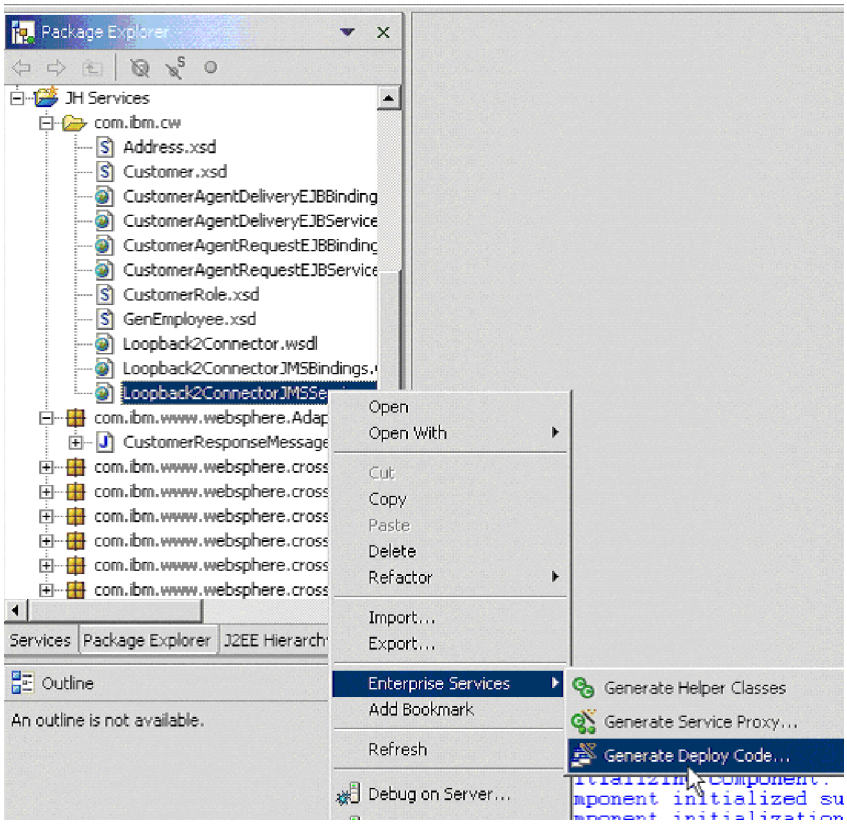
- .xsd ファイルを Java Bean として表現する helper クラス、および .wsdl ファイルに指定されている内容に従って XML を Java に変換するフォーマット・ハンドラー Java Bean

上記の各作業については、以降の各トピックで説明します。

EJB プロジェクトの作成

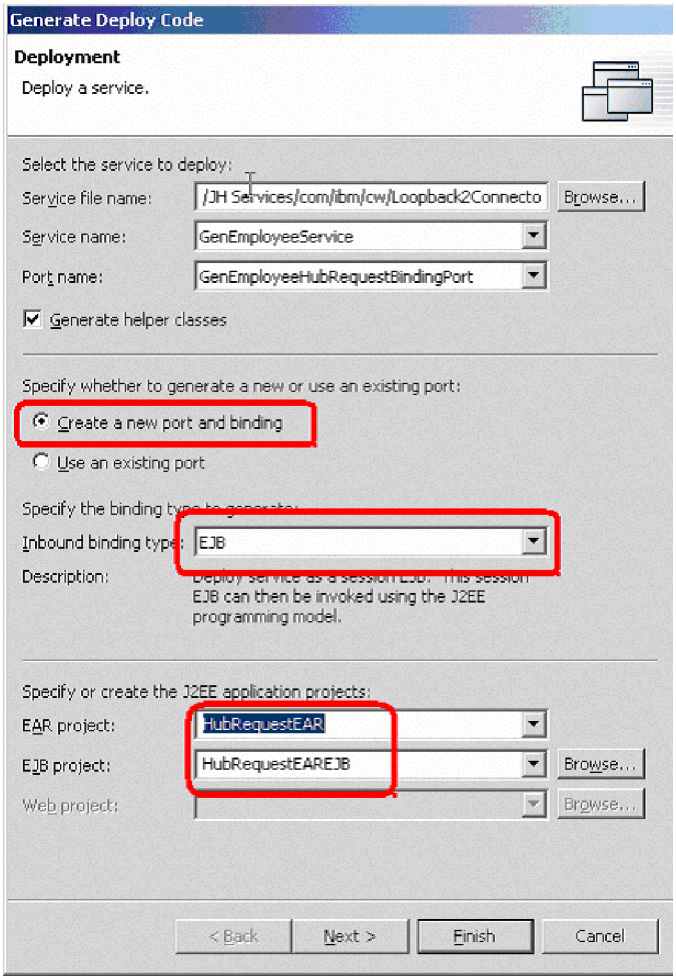
このステップでは、Session Bean と、この Bean を EJB サービスおよびバインディングとして記述した .wsdl ファイルを作成します。

1. パッケージ・エクスプローラー・ビューで、JMS サービスの .wsdl ファイルを選択します。(サービス・プロジェクトは、4 つの対話パターンのすべてで同じ内容です。そのため、ここで選択するファイルは、エージェント・デリバリー対話パターン用にサービス・プロジェクトを作成したときに生成された JMS サービス・ファイルと同じものにすることができます。)
2. 選択したファイルを右マウス・ボタンでクリックし、「エンタープライズ・サービス (Enterprise Services)」>「配置コードの生成 (Generate Deploy Code)」を選択します。「配置コードの生成 (Generate Deploy Code)」ダイアログが表示されます。

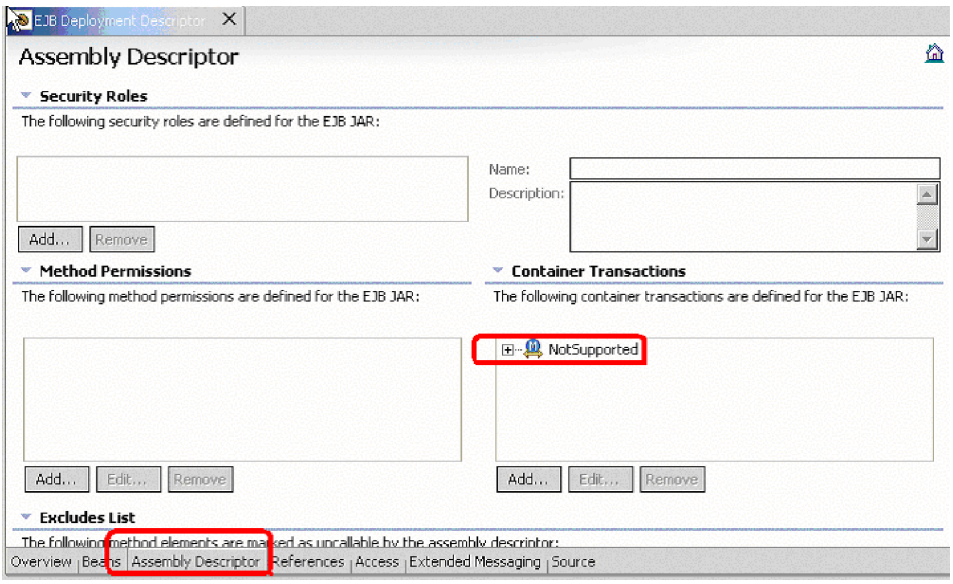


3. 「ポート名」ドロップダウンで、目的の対話パターンのポート名を選択します。

4. 「*新規ポートおよびバインディングを作成 (Create a new port and binding)*」を選択します。



5. 「*インバウンド・バインディング・タイプ (Inbound binding type)*」ドロップダウンで、「EJB」、「JMS」、または「SOAP」を選択します。選択した内容は、生成される WSDL ファイルに影響します (EJB アクセス用に EJB バインディングおよびサービスに関するファイルが生成されます)。
6. 「次へ」を選択します。「インバウンド・サービス・ファイル (Inbound Service Files)」ダイアログが表示されます。
7. 「インバウンド・サービス・ファイル (Inbound Service Files)」ダイアログで、必要に応じて編集を行い、ポート名とバインディング名の両方に分かりやすいパッケージ名を指定します。
8. JNDI ルックアップ名を変更する必要がある場合は、「次へ」を選択します。その必要がない場合は、「完了 (Finish)」を選択します。
9. 配置記述子エディターを使用して、ハブ片方向対話パターンで行った方法と同じ方法で、キュー名およびキュー接続ファクトリー名のマッピングを行います。ただし、ハブ要求対話パターンを正常に実行するには、メッセージの送信および受信を同じトランザクションの一部にはしないでください。これを簡単に解決するには、次の図に示すように、EJB の配置記述子にある EJB のトランザクションを使用不可にします。



10. ハブ要求を表現する EJB が完成しました。この EJB は、配置することも、別の WebSphere Application Server のコンポーネントとして使用することもできます。

ここで説明したステップを完了した後は、テスト用のアプリケーション・クライアント・プロジェクトを作成することができます (オプション)。ハブ片方向対話パターンを対象にこの作業について説明したステップを参照してください。

対話パターン内でのビジネス・オブジェクト初期化の要件

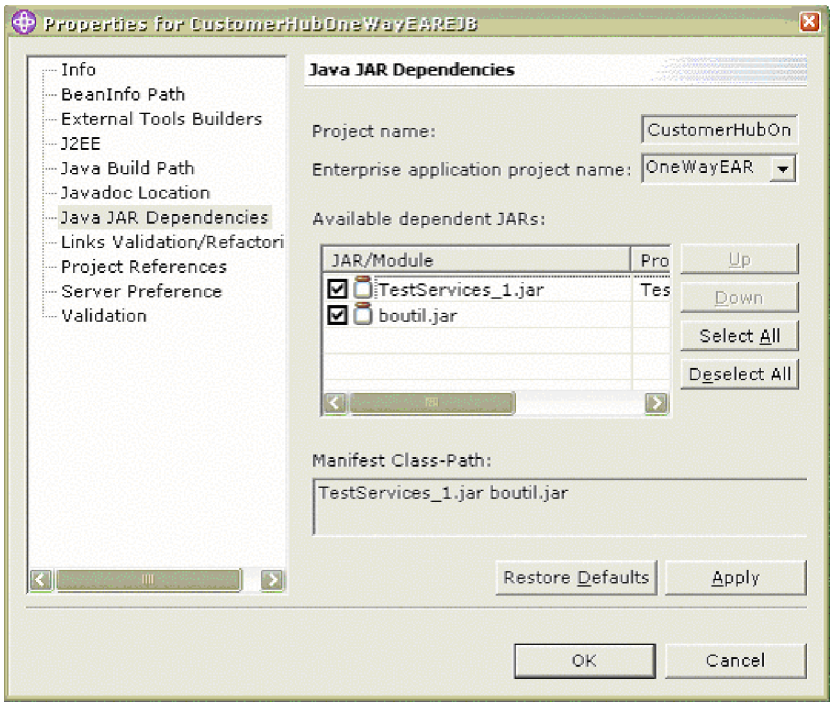
本セクションのトピックでは、対話パターン内でビジネス・オブジェクトを使用するために必須の要件について説明します。

ビジネス・オブジェクト初期化ライブラリーの追加

ビジネス・オブジェクトを作成する EJB には、BusinessObject 初期化ライブラリー・ルーチンが必要です。このライブラリー・ルーチンを含む JAR ファイル (boutils.jar) は、Enterprise Application Archive プロジェクトの一部を構成し、EJB モジュールの依存ファイルに指定されている必要があります。これを行うためのステップは、次のとおりです。

1. Business Integration パースペクティブの J2EE 階層ビューで、Enterprise Application Archive アプリケーションを選択します。右マウス・ボタンでクリックし、「インポート」を選択します。
2. 「インポート」ダイアログで、アダプターに付属している boutils.jar ファイルの格納先に移動し、このファイルをインポートします。
3. 新しくインポートしたファイルが、エンタープライズ・アプリケーションのフォルダー階層内に表示されていることを確認します。
4. インポートした JAR ファイルを、目的のユーティリティー・ライブラリーを使用する EJB モジュール (選択した Enterprise Application Archive に含まれるもの) の依存ファイルのリストに追加します。EJB プロジェクトを選択し、ポップ

アップ・メニューから「プロパティー」を選択します。次に、リストから「Java JAR 依存関係 (Java JAR Dependencies)」を選択します。



5. boutil.jar のボックスにチェックマークを付けます。
6. 「OK」を押します。

これで、boutils.jar ファイルがエンタープライズ・アプリケーションの一部になり、Enterprise Application Archive ファイルに組み込まれるようになりました。

ビジネス・オブジェクトの処理における予約値

実行時に、CxIgnore 属性値または CxBlank 属性値を含むビジネス・オブジェクトが送信元アダプターから WebSphere Application Server のブローカー実装に送信されると、CwXML データ・ハンドラーは、ブローカーに渡す XML インスタンス文書の中で、これらの属性値を予約値で置き換えます。また、そのようなビジネス・オブジェクトがブローカーで処理されて宛先アダプターに XML メッセージとして戻されると、宛先アダプターのデータ・ハンドラーによって、CxIgnore 値と CxBlank 値が復元されます。この一連の処置は、ブローカーの Java オブジェクトでビジネス・オブジェクトが処理されるときに、それらの属性にデフォルト値ゼロが割り当てられるのを防止するために必要な処置です。

次の表に、値 CxIgnore と CxBlank の代わりに使用される予約値をデータ型ごとに示します。

データ型	CxIgnore	CxBlank
int	Integer.MIN_VALUE	Integer.MAX_VALUE
float	Float.MIN_VALUE	Float.MAX_VALUE
double	Double.MIN_VALUE	Double.MAX_VALUE
string、date、longtext	″CxIgnore″	

注: データ型が boolean の場合、使用可能な値は「true」と「false」のいずれかです。この制限のため、データ型が boolean の場合には、CxIgnore や CxBlank を値として設定することはできません。データ型が boolean の場合の対処手順は、属性のタイプを string データ型として定義してから、「CxIgnore」または「CxBlank」を値として設定するという手順になります。

アダプター・エージェントから送信されたビジネス・オブジェクトは、前記の表の値で開始されます。ただし、ハブ片方向対話パターンやハブ要求対話パターンなどのユーザー・アプリケーションで作成されたビジネス・オブジェクトは、明示的に開始する必要があります。これを行うには、以下の部分的なサンプル・コードが示すように、ビジネス・オブジェクトの作成のたびに BusinessObjectUtilities.initializeBO(<BO インスタンス>) を呼び出します。

```
TestCustomerElement customer = new TestCustomerElement();
        AddressElement address = new AddressElement();

                customer.setAddress(address);
        BusinessObjectUtilities.initializeBO(customer);

        customer.setCustomerId("2424234");
        address.setCountry("USA");
```

ビジネス・オブジェクト属性の設定を行う場合は、あらかじめ、すべてのビジネス・オブジェクトを作成し、それらのビジネス・オブジェクトの親/子構造を作成して、親ビジネス・オブジェクトに対し初期化ルーチンを呼び出しておく必要があることに注意してください。

トランザクションのサポート

各対話パターンでは、以下の条件付きでトランザクション動作がサポートされます。

ハブ要求パターンでは、メッセージの送信および応答の待機と受信は、同一のトランザクションに属していてはいけません。同一のトランザクションを使用すると、J2EE コンポーネントが無限停止状態になります。これは、メッセージを送信前に受信することは不可能であるためです (このとき、メッセージの送信は、トランザクションがコミットされない限り発生しません)。

エージェント側から開始される対話パターン (エージェント・デリバリーおよびエージェント要求) でトランザクションを使用する場合には、メッセージ・リスナーが使用不可になるのを防止するために、特別な対策を講じる必要があります。MDB がエージェントから発信されたメッセージを受信した後で、その MDB 自体、またはその MDB から呼び出された別の WebSphere Application Server コンポーネントで例外がスローされると、トランザクションはロールバックされ、メッセージはキューに戻されて再配信されます。このエラーが続いた場合、再配信は、事前定義された回数繰り返されてから停止するか (このとき、リスナーは使用不可になります)、無限に繰り返されるかのどちらかです。どちらになるかは、リスナー・ポートの設定によって決まります。トランザクションを使用する場合にこのような状態に陥るのを防止するには、再配信されたメッセージを検出して処理するユーザー・コードを MDB に追加します。次のサンプルは、再配信されたメッセージを MDB で検出するための方法を示しています。

```
    public void onMessage(javax.jms.Message msg) {
            try {
// -> new code start
```

```
                    if(msg.getJMSRedelivered()){
                           System.out.println("Message redelivered");
                           // User code to handle redelivered message
                       }
                       else{
                           System.out.println("Message delivered first time");
                         // First time delivery, unless user code for
                           // redelivery returns, it should invoke
    executeOperation(msg)
                        }
// <- new code end
                executeOperation(msg);
          }
        catch (WSIFException e) {
                e.printStackTrace();
          }
        catch (javax.jms.JMSException e) {
                        e.printStackTrace();
          }
    }
```

第 9 章 ビジネス・インテグレーション・システムの管理

本章では、WebSphere Business Integration Adapters のために実行する必要がある、管理用タスクについて説明します。

本章は、次の各セクションから構成されます。

- 『コネクターの始動』
- 118 ページの『コネクターの停止』
- 121 ページの『ログ・ファイルおよびトレース・ファイルの管理』
- 130 ページの『Log Viewer を使用してコネクター・メッセージを表示する方法』

コネクターの始動

コネクターの始動方法は、動作環境が UNIX と Windows のいずれのシステムであるかによって異なります。デフォルトでは、コネクター構成ファイル内の設定値がコネクター・プロパティーの値となります。ただし、コネクターの構成ファイルで指定された一部のコネクター・プロパティーは、実行時にオーバーライドすることができます。163 ページの『付録 C. コネクター始動オプション』では、コネクターの構成ファイル内の設定済みプロパティーをオーバーライドするために、使用するオプションについて紹介しています。

注: 必ず、72 ページの『コネクターの始動ファイル、ショートカット、および環境変数の構成』での指示に従ってから、コネクターを始動してください。

Windows の場合

以下の方法により、コネクターを始動します。

- ショートカットを作成済みの場合は、デスクトップ上でそのショートカットをダブルクリックする。
- **「プログラム」>「IBM WebSphere Business Integration Adapters」>「アダプター」>「コネクター」**の順に選択し、そこで対象のコネクターを選択する。プログラム名は、デフォルトでは「IBM WebSphere Business Integration Adapters」となっていますが、それをカスタマイズすることもできます。
- System Manager の Adapter Monitor パースペクティブからコネクターを始動できます。詳しくは、126 ページの『アダプターの状態の変更』 を参照してください。
- DOS ウィンドウで次の処理を実行します。
  - 大半のコネクターでは、以下のコマンドを入力します。

    ```
    start_connName connectorName brokerName -cConfigFile
    ```

    ここで、

    connName および *connectorName* はコネクター名、*brokerName* は WebSphere Message Broker の名前で、*configFile* はコネクターの構成ファイルの絶対パス名を指定しています。

コネクターの始動オプションは、コネクターのショートカット・プロパティーを編集するか、またはコネクターの start_*connName*.bat ファイルを直接編集すれば、その内容を変更することができます。

注: 大半のコネクター、例えば Java ベースのすべてのコネクターや IBM が提供するほぼすべてのコネクターでは、上記の構文を使用します。ご使用のコネクターがこの構文を使用するかどうかは、ディレクトリー *ProductDir*/connectors 内を調べると確認できます。start_connName という名前のスクリプトが存在する場合は、上記の構文を使用します。C++ 言語で記述された一部のコネクターでは、代わりに後のセクションで説明する構文を使用する必要があります。このディレクトリー内にコネクターに関連付けられた start_*connName* スクリプトが存在しない場合は、後述する構文を使用します。

– 上記の start_*connName* 構文を使用できない C++ コネクター (一部の IBM コネクターおよびカスタム・コネクター) の場合、コネクターがインストールされているディレクトリーにナビゲートします。デフォルトでは、このディレクトリーは *ProductDir* です。下記のコマンドを入力します。

```
start_connector connectorName brokerName -cConfigFile
```

ここで、

connectorName はコネクター名、*brokerName* は WebSphere Message Broker の名前で、*configFile* はコネクターの構成ファイルの絶対パス名を指定しています。

Java コネクターの始動オプションは、コネクターのショートカット・プロパティーを編集するか、またはコネクターの start_connector.bat ファイルを直接編集すれば、その内容を変更することができます。

UNIX の場合

以下の方法により、コネクターを始動します。

1. *ProductDir*/bin ディレクトリーにナビゲートする。
2. 下記のコマンドを発行してコネクター・マネージャー・スクリプトを実行する。

```
connector_manager_connName -start
```

ここで、

connName はコネクター名です。

このコネクター名は、大文字や小文字およびスペルでも、*ProductDir*/bin/connectors の配下にあるコネクターのサブディレクトリー名と一致しなければなりません。

コネクターの停止

Windows の場合

Windows からコネクターを停止するには、以下に挙げるいくつかの方法があります。

- System Manager の Adapter Monitor パースペクティブからコネクターを停止できます。詳しくは、126 ページの『アダプターの状態の変更』 を参照してください。
- Adapter Monitor からコネクターを停止できない場合、コネクター・タイプに応じたコンソール・ウィンドウで「Q」と入力し、Enter キーを押します。これによって、コネクターのプロセスが強制終了されます。

UNIX の場合

UNIX システムの場合は、以下のコマンドを使用します。

1. *ProductDir*/bin ディレクトリーにナビゲートします。
2. 以下のように入力します。connector_manager_*connName* - *option*

ここで、*option* は以下のオプションのいずれかです。

表 10. connector_manager コマンド停止オプション

オプション	説明
- stopgraceful	コネクター・エージェント側から Java ユーティリティーを呼び出して、コネクター・エージェントを停止します。ただし、自動再始動機能が有効になっている場合、コネクター・エージェントは自動的に再始動します。
-stop	コネクター・エージェントを停止する Java ユーティリティーを呼び出します。自動再始動機能が有効な場合でも、コネクター・エージェントは強制的にシャットダウンされます。
-kill	コネクター・エージェント用のオペレーティング・システム・プロセスを強制終了します。他の方法でコネクターを停止できなかった場合に使用します。

複数のコネクター・インスタンスの作成

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

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

新規ディレクトリーの作成

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

```
ProductDir¥connectors¥connectorInstance
```

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

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

```
ProductDir¥repository¥connectorInstance
```

ビジネス・オブジェクト定義の作成

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

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

   ```
   ProductDir¥repository¥initialConnectorInstance
   ```

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

コネクター定義の作成

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

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

始動スクリプトの作成

始動スクリプトは以下のように作成します。

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

   dirname

2. この始動スクリプトを、 119 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. 始動スクリプトのショートカットを作成します (Windows のみ)。
4. 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。

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

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

WebSphere MQ キューからのメッセージのクリア

WebSphere Business Integration Adapters にはサンプル・バッチ・ファイルが用意されています。サンプル・バッチ・ファイルを使用して、ビジネス・インテグレーション・システムの WebSphere MQ キューからのメッセージをクリアすることができます。ビジネス・インテグレーション・システムに問題があることから、処理のためにメッセージを除去できない場合は、キューのクリアが必要になることがあります。

WebSphere MQ キューからのメッセージをクリアするには、バッチ・ファイル `clear_mq.bat` (Windows) または `clear_mq` (UNIX) を実行します。これらのバッチ・ファイルは、*`ProductDir`*`¥templates` ディレクトリーにあります。このバッチ・ファイルにより `crossworlds_mq.tst` ファイルに指定されたキューからメッセージがクリアされます。 `crossworlds_mq.tst` の編集についての詳細は、61 ページの『WebSphere Business Integration Adapter バッチ・ファイルを使用して WebSphere MQ キューを構成する方法』を参照してください。

ログ・ファイルおよびトレース・ファイルの管理

次の各ツールには、メッセージのロギングおよびトレースを構成および表示するための、グラフィカル・ユーザー・インターフェース (GUI) が提供されています。

- コネクターのロギングおよびトレースをセットアップまたは変更する Connector Configurator。
- ログ・ファイルおよびトレース・ファイルを表示する Log Viewer。

Log Viewer を使用してログを表示するほかに、テキスト・エディターでログを開くこともできます。

始動時に、コネクターにより一時的ログ・ファイルが生成されます。このファイルには、コネクター・フレームワークに渡されるコネクター・プロパティーおよびビジネス・オブジェクト定義など、始動時にログとして記録されるメッセージがすべて格納されます。ファイル名は `broker_name_connector_name_tmp.log` で、*`ProductDir`* ディレクトリーに書き込まれます。コネクターを実行すると、標準のコネクター構成プロパティーに構成されたとおりに、ロギングおよびトレースが処理されます。

コネクターの構成ファイルで指定されたロケールの UTF-8 エンコードを使用してロギング・メッセージおよびトレース・メッセージを書き込んでいる間に、ロケールに依存しない形式で時刻情報が書き込まれます。

コネクターの始動時にログ・ファイルが存在していない場合は、コネクターの構成ファイルで指定されたロケーションに、ログ・ファイルも作成されます。ログ・ファイルが存在している場合は、新規のエントリーが追加されます。コネクターのロ

グ・ファイルにサイズの上限値が設定されていない場合、そのサイズは最近に管理されてからの経過時間およびシステムを経由したトランザクション量によって決まります。コネクターのログ・ファイルのサイズが制限なしのままでそのファイルを構成すると、ファイルのボリュームが増加し続けて、ファイルが開けなくなったりディスク・スペースが不足したりするような、事態を招くこともありえます。

トレースが使用可能である場合、トレース・ファイルは、まだ存在していなければ始動時に作成されます。トレース・ファイルのサイズは、コネクターのログ・ファイルの場合と同じ方法で管理することにより、ボリュームの肥大化によるトラブルを回避する必要があります。

70 ページの表 8 では、コネクターがロギングおよびトレース情報を格納するために使用するファイルが、紹介されています。

ログ・ファイルおよびトレース・ファイルを管理するには、Connector Configurator を使用して以下の作業を実行します。

- ログ・ファイルおよびトレース・ファイルの、サイズの上限値を指定する。
- サイズの上限値に達したら、ファイルを自動的にアーカイブする。
- 保存するアーカイブ・ファイルの数を指定する。

Connector Configurator を使用した、これらのオプションの設定に関する詳細については、 69 ページの『ロギングおよびトレース・オプションの構成』を参照してください。

ログ・ファイルおよびトレース・ファイルのアーカイブ・ロギング

アーカイブ・ロギングの使用が可能である場合、コネクターのログ・ファイルまたはトレース・ファイルのボリュームが最大値に達するたびに、新規のアーカイブ・ファイルに名前が変更されます。アーカイブ・ファイルの名前は、オリジナルのログ・ファイルまたはトレース・ファイル名から派生しており、その名前には *_Arc_number* が挿入されています。

例えば、5 つのアーカイブ・ファイルが使用され、ログ・ファイルの名前が `Connector.log` である場合は、次のようになります。

- 最初に作成されるアーカイブの名前は `Connector_Arc_01.log` です。
- 新規のログ・ファイルが満杯になると、`Connector_Arc_01.log` の名前が *Connector*`_Arc_02.log` に変更されます。
- 新規のログ情報は再度 `Connector_Arc_01.log` に格納され、5 つのアーカイブ・ファイルになるまで、このように循環します。
- 新規のログ・ファイルを作成するときに、アーカイブ・ファイルがすでに 5 つ存在している場合は、最も古いアーカイブ・ファイル (5 番) が削除されます。次に、残りのアーカイブ・ファイルは名前が変更されて番号がアップし、アーカイブ数と構成した数が合うようになっています。 123 ページの図 19 に、この構成を使用したファイルの循環状況を示します。

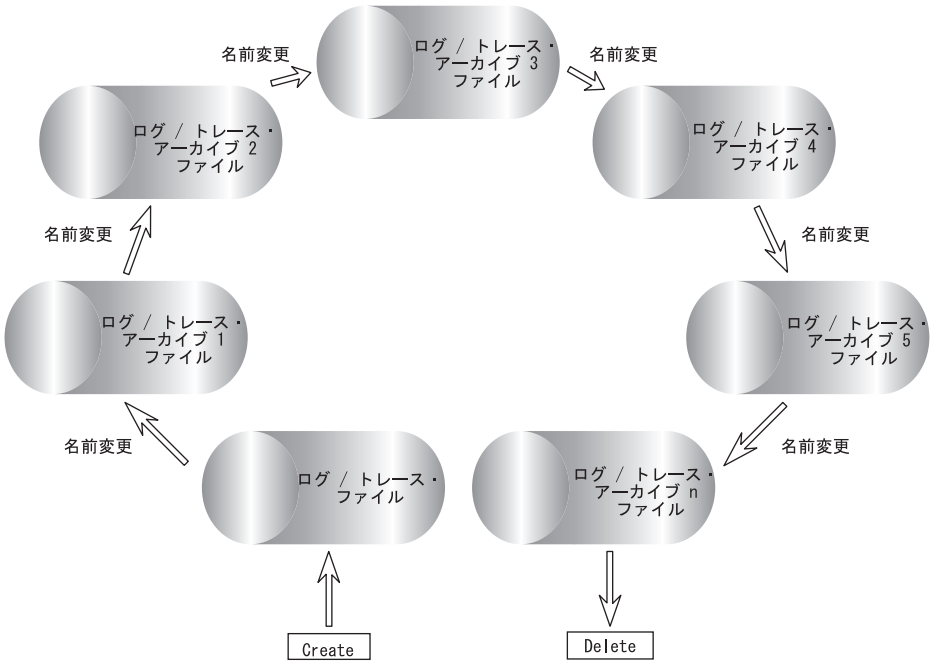


図 19. 循環アーカイブ・ロギング

詳細については、 69 ページの『ロギングおよびトレース・オプションの構成』の構成作業を参照してください。

その他のファイルの管理

ログ・ファイルおよびトレース・ファイルはアーカイブ・ロギングで管理できますが、各アプリケーションに固有なその他のログ・ファイルについては、手作業で管理する必要があります。これらのファイルの多くは、存在していない場合は実行時に作成されます。既存のファイルである場合は、そこに新規の情報が追加されます。

ファイル管理手順は任意に使用することができますが、IBM では次のログ・ファイル管理を定期的に行なうことをお勧めします。

- ファイル名に日付を付加して名前を変更する。
- ファイルをアーカイブ・ディレクトリーに移動する。

Adapter Monitor および障害キュー・マネージャーの使用

Adapter Monitor は、アダプターの状態の変更に使用できるパースペクティブです。また、障害キューで受信した失敗イベントを、障害キュー・マネージャーを通じて処理することができます。System Manager の使用についての詳細は、「*IBM WebSphere InterChange Server システム管理ガイド*」を参照してください。

注: このセクションでは、「アダプター」という用語は、これ以外の WebSphere Business Integration Adapters の資料における「コネクター」と同じ意味です。このセクションで「アダプターの状態の変更」に言及している場合は、「コネクターの状態の変更」のことを指しています。

Adapter Monitor パースペクティブ

Adapter Monitor パースペクティブを使用すると、アダプターを管理できます。また、サブミットされたイベントの処理時にエラーが発生した場合には、障害キュー・マネージャー・パネルからメッセージを再サブミットすることもできます。

Adapter Monitor は、JMS と共に使用するように構成されているアダプターに対してのみ使用できます。

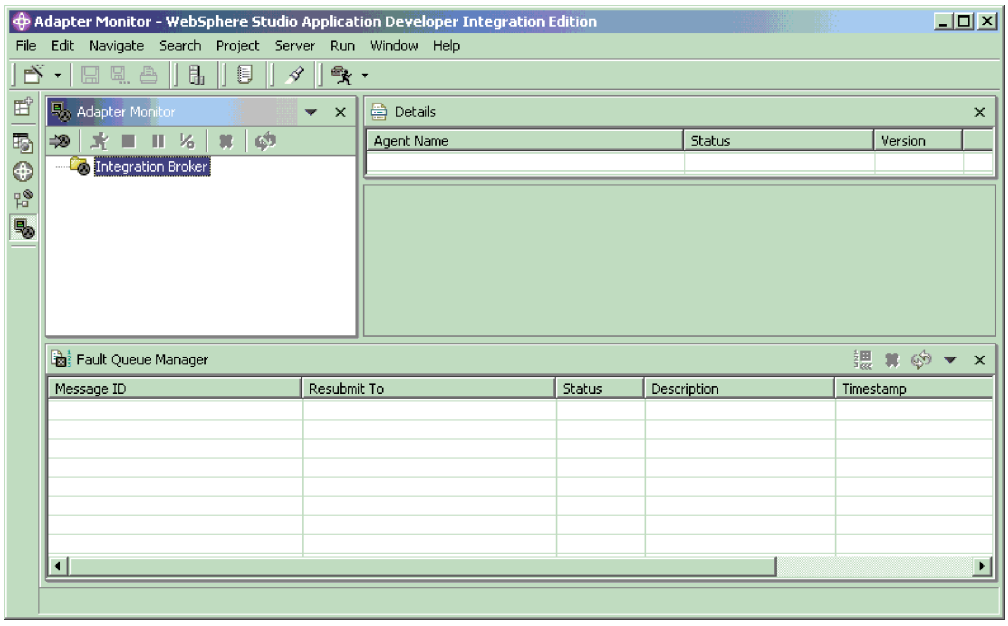
注: Adapter Monitor は、Windows 環境で稼働します。UNIX システムにインストールされたアダプターに対して Adapter Monitor を使用するには、リモート・キューを作成する必要があります。

Adapter Monitor の開始

Adapter Monitor は、次のいずれかの方法で開くことができます。

- WebSphere Studio Application Developer Integration Edition のメニュー・バーから、「ウィンドウ」>「パースペクティブを開く」>「Adapter Monitor」を選択します。
- System Manager パースペクティブで、WebSphere Application Server プロジェクトのフォルダーを展開し、コネクター定義のアイコンを右マウス・ボタンでクリックして、ポップアップ・ダイアログから「Adapter Monitor」を選択します。

どちらの方法を使用した場合も、次の図に示す「Adapter Monitor」ウィンドウが表示されます。



Adapter Monitor の設定の変更

Adapter Monitor がアダプターの状態をポーリングする間隔を決定する設定や、障害キュー・マネージャーに表示されるメッセージの数を決定する設定は、変更することができます。これらの設定を変更するには、WebSphere Studio Application Developer Integration Edition のメニュー・バーから、「ウィンドウ」>「設定」>「Adapter Monitor の設定」を選択し、次のように値を設定します。

- Adapter Monitor ビュー

  アダプター状況のポーリングが行われてからその次に行われるまでの間の経過秒数を表す数値を入力します。
- 障害キュー・マネージャー

  障害キュー・マネージャーに表示するメッセージの数の最大値を表す数値を入力します。

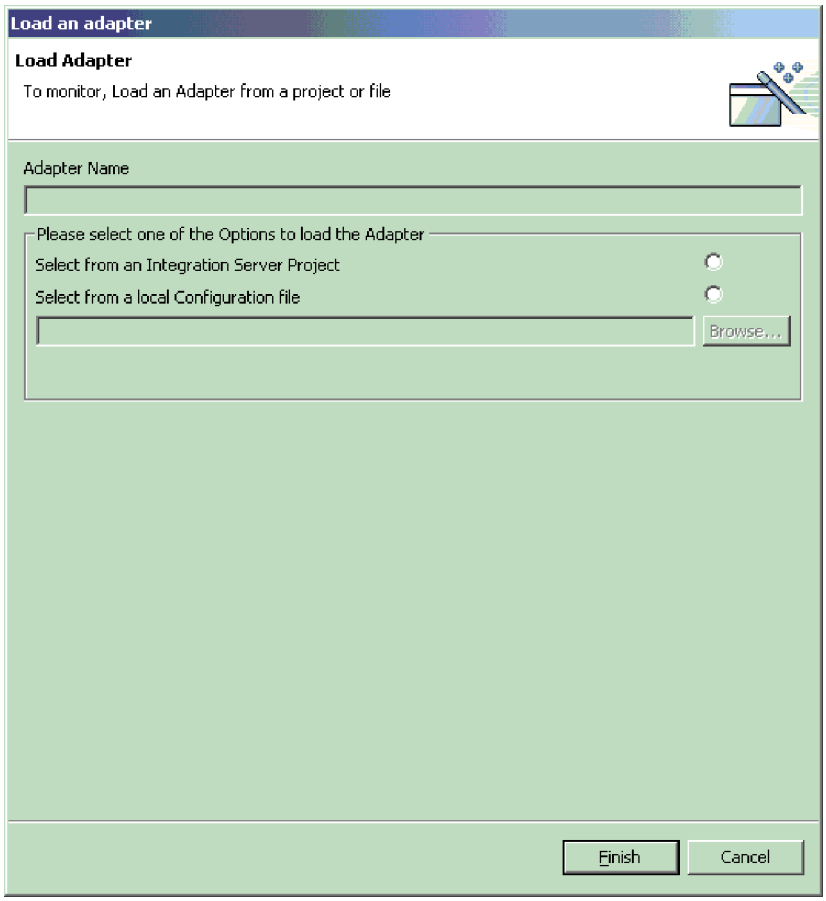
「適用」または「OK」を選択します。

アダプターのロード

Adapter Monitor を使用するには、まずキュー・マネージャーを始動します。次に、アダプターを始動し、Adapter Monitor ビューにそのアダプターをロードします (Adapter Monitor ビューにアダプターをロードしてから、アダプターを始動してもかまいません)。

アダプターを始動するには、コネクター始動スクリプト、ショートカット、Visual Test Connector などの、そのアダプターを始動するためにセットアップした手段を使用します。

Adapter Monitor にアダプターをロードするには、「統合ブローカー」を右マウス・ボタンでクリックし、「アダプターのロード」を選択します。次の図に示すダイアログが表示されます。



アダプターの構成ファイルと統合プロジェクトのどちらからアダプターをロードするかを選択することができます。

- プロジェクトをロード元として選択した場合、ダイアログには、System Manager で現在有効なユーザー・プロジェクトが表示されます。プロジェクト内のコネクターを選択し、「完了 (Finish)」を選択します。(有効なユーザー・プロジェクトがない場合には、何も表示されません。この場合、新しいユーザー・プロジェクトを作成するか、構成ファイルをロード元にします。)
- ファイルをロード元として選択した場合、参照ボックスが開きます。アダプターの構成ファイル (通常、構成ファイルの拡張子は *.cfg ですが、別の拡張子になっている場合もあります) の格納先に移動してこのファイルを選択し、「保管」を選択します。選択した構成ファイルの名前が、ボックスに表示されます。「完了 (Finish)」を選択します。Adapter Monitor にアダプターがロードされ、そのアダプターの現在のデータが表示されます。

Adapter Monitor の表示

アダプターをロードすると、左上のパネルの「統合ブローカー」フォルダーの下に、そのアダプターを表すアイコンが表示されます。このアイコンは、アダプターの現在の状態を示しています。Adapter Monitor にアダプターをロードする操作そのものによって、アダプターの状態が変化することはありません。アダプターのロードが完了すると、アダプターの状態を変更するためのアクションを Adapter Monitor から実行することができます。

Adapter Monitor の表示は、Adapter Monitor ビューの設定で指定したポーリング時間間隔に従って、定期的に更新されます。ただし、ツールバーまたはメニュー・バーの「最新表示」ボタンを選択すれば、いつでもすぐに更新することができます。

Adapter Monitor の右上のパネルは、「詳細」パネルです。「詳細」パネルには、アダプターの名前、状況、およびバージョンが表示されます。

Adapter Monitor の下部パネルは、障害キュー・マネージャー・ビューになっています。このビューには、キュー・マネージャーの障害キューへ発信されたメッセージが表示されます。この障害キュー・マネージャー・ビューでは、イベント・フローの失敗の結果として障害キューに置かれたメッセージを、再サブミットするか、削除することができます。

アダプターの状態の変更

Adapter Monitor を使用すると、アダプターの状態をモニターして変更することができます。アダプターの状態とは、アダプターによって実行されている (または実行されていない) 処理のことを指します。

注: アダプターの「状態」が存在すると、アダプターが始動している状態であることが推定できます。始動していないアダプターやシャットダウン後に再始動していないアダプターには状態が存在しないため、Adapter Monitor の動作による影響はありません。

通常、アダプターは、次の 2 種類の処理を実行します。

- イベント通知のポーリング

アダプターは、そのアプリケーションのイベント・ストアを複数のイベントに対してポーリングし、これらのイベントをビジネス・オブジェクト・メッセージとして統合ブローカーに送信します。

- 要求処理

  アダプターは、統合ブローカーからアプリケーションに送信された要求ビジネス・オブジェクトを受信します。

アダプターの状態	要求処理	ポーリング
Active	あり	あり
一時停止	あり	なし
非アクティブ	なし	なし

アダプターの状態を変更するには、そのアダプターを表すアイコンを右マウス・ボタンでクリックし、次のいずれかを選択します。

- アクティブ化

  アダプターを「一時停止」状態または「非アクティブ」状態から「アクティブ」状態へ変更します。

- 非アクティブ化

  アダプターを「一時停止」状態または「アクティブ」状態から「非アクティブ」状態へ変更します。

- 一時停止

  アダプターを「アクティブ」状態状態から「一時停止」状態へ変更します。

- シャットダウン

  アダプターを停止します。この動作により、コネクター始動スクリプトが終了し、アプリケーションとの接続が切断されて割り当てられているリソースがすべて解放されます。アダプターは再始動するまでシャットダウンしたままです。アダプターは、Adapter Monitor から再始動することはできません。

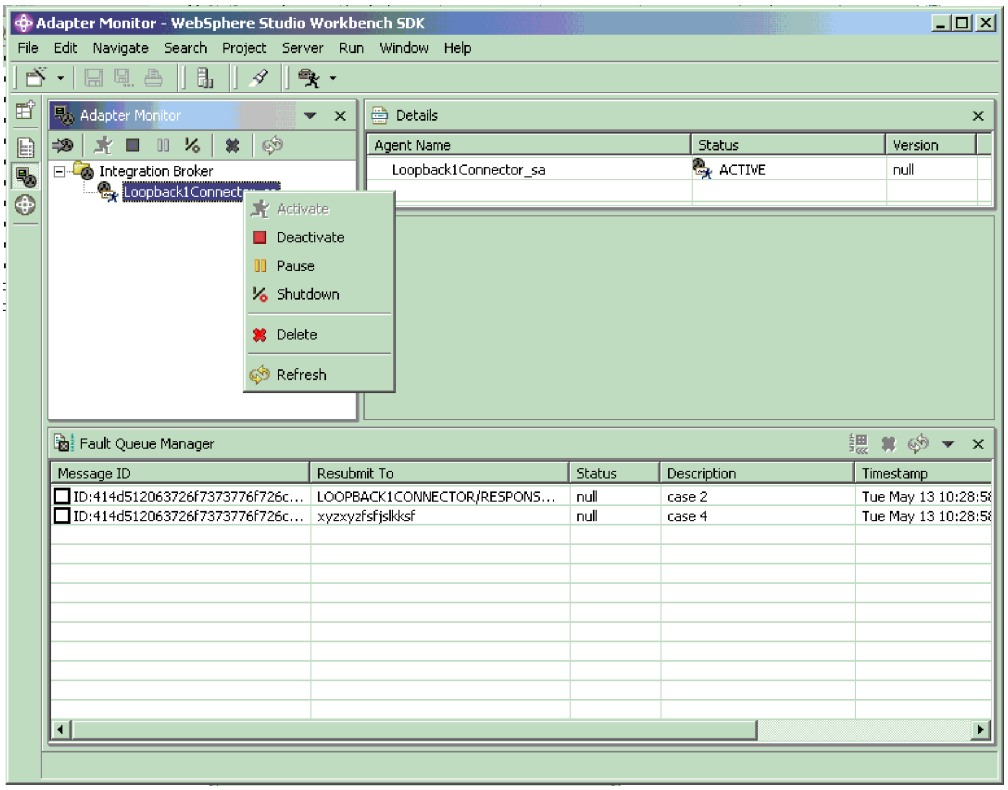
次の 2 つのコマンドは、コネクターの処理に直接の影響を与えることはありませんが、Adapter Monitor がコネクターについて表示する内容には影響があります。

- Delete

  Adapter Monitor からこのアダプターの構成情報を削除します。このコマンドは、アダプターを変更することも、アダプターの状態を変更することもありません。Adapter Monitor から項目を削除するのみです。

- 最新表示

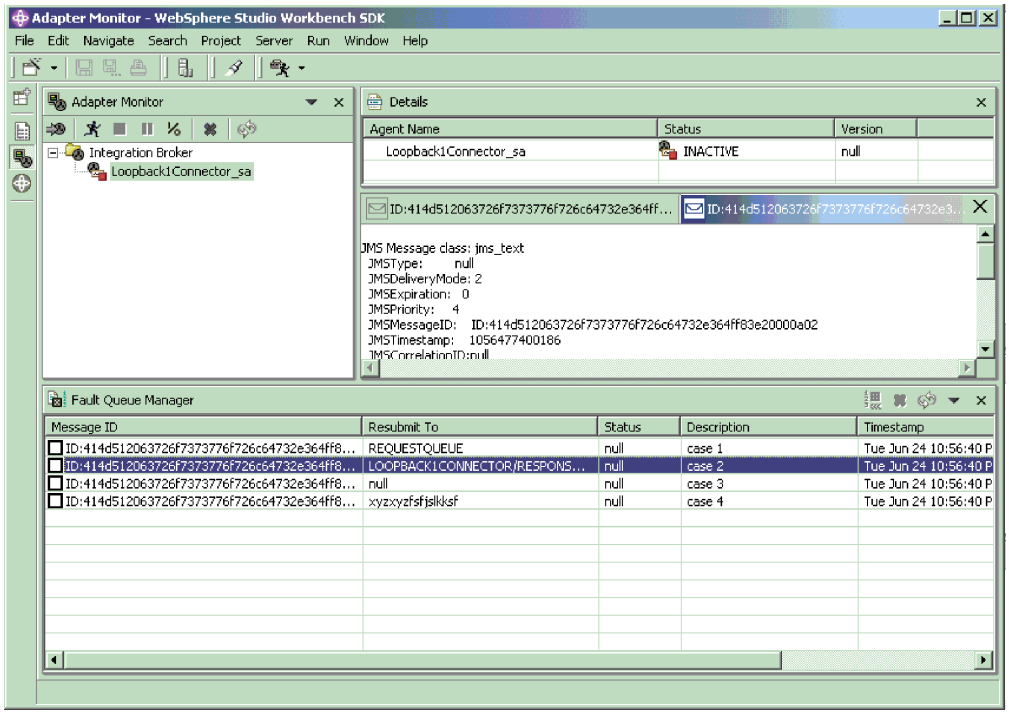
  コネクター・エージェントの現在の状態を読み取る GETSTATUS コマンドを起動します。



障害キュー・マネージャー・ビューの使用

障害キュー・マネージャー・ビューには、障害キューで受信した失敗イベントが表示されます。このビューには、設定により指定した数の失敗メッセージが、受信順にリスト表示されます。

リスト表示されている障害キュー・メッセージに関するキュー・マネージャーからの情報を表示するには、メッセージ・フィールドをダブルクリックします。次の図に示すように、そのメッセージの情報がエディター・ビュー・パネルに読み取り専用で表示されます。



失敗イベントの処理

障害キュー・マネージャーには、コンテナー管理イベント用の 2 種類の対話パターン (ハブ要求 (同期イベント・デリバリー) 対話パターンとエージェント・デリバリー (非同期イベント・デリバリー) 対話パターン) から受信した失敗イベント・メッセージが表示されます。また、このキュー・マネージャーを使用すると、表示されたメッセージを処理することができます。

イベント・メッセージをキューから削除するか、イベント・メッセージの再サブミットを試みることができます。

イベントを再サブミットするには、そのイベントの「メッセージ ID」フィールドのチェック・ボックスにマークを付け、右マウス・ボタンでクリックして「再サブミット」を選択します。

Adapter Monitor が、選択されたイベントの再サブミットを試みます。再サブミットが成功すると、イベントは障害キュー・マネージャー・ビューから除去されます。

「ResubmitTo」フィールドの値がヌルのメッセージを再サブミットすることはできません。イベントを再サブミットしようとして失敗するのは、メッセージ自体が無効であるか、ResubmitTo キューが無効または使用不可であるためです。

メッセージが無効になるのは、そのメッセージに無効な JMS タイプが含まれているか、そのメッセージをビジネス・オブジェクトに変換できないためです。どちらに該当する場合でも、無効なメッセージを再サブミットしようとして失敗したときには、その無効なメッセージに関するエラーが障害キュー・マネージャーによって表示されます。「OK」を選択すると、そのメッセージは再サブミットされなくな

り、障害キューのビューから除去されます。「OK」を選択しないでダイアログを閉じると、そのメッセージは障害キューのビューに残されます。

メッセージ自体は有効であるにもかかわらず、メッセージの ResubmitTo キューがヌル、無効、または使用不可であるために再サブミットに失敗した場合は、「再サブミット」ダイアログが表示され、ここでメッセージの値を確認できます。キューにメッセージを残す (「キャンセル」を選択する) か、キューからメッセージを削除する (「OK」を選択する) かを選ぶことができます。

Log Viewer を使用してコネクター・メッセージを表示する方法

Log Viewer を使用すると、コネクターのログ・ファイルおよびトレース・ファイルに格納されているメッセージを閲覧することができます。出力表示内容のソートおよびフィルター、さらにファイルの印刷、保存、電子メール送信を行なうこともできます。ロギングおよびトレース・オプションと、生成されたファイルのロケーションは、コネクターの構成ファイル内のプロパティーとして指定されています。

注: Log Viewer は、Windows 2000 マシン上でのみ動作します。Log Viewer を使用して UNIX ログ・ファイルを構成または表示するには、UNIX マシンから Windows マシンにログ・ファイルをコピーして、Windows マシンで表示します。

Log Viewer を始動するには、「スタート」メニューの「ファイル名を指定して実行」を選択し、参照によって `LogViewer.exe` ファイルを探し出して、実行します。「ファイル」メニューの「開く」オプションを使用して、ログ・ファイルをブラウズします。この方法を使用する代わりに、`LogViewer.exe` への Windows ショートカットを作成して使用してもかまいません。

「Log Viewer」メニュー・オプションを使用すると、以下の各タスクが実行できます。

- 131 ページの『Log Viewer の設定の指定』
- 134 ページの『メッセージの表示法の変更』
- 136 ページの『Log Viewer 表示出力の制御』

サンプル・ログ・ファイルを表示している Log Viewer を、131 ページの図 20 に示しています。

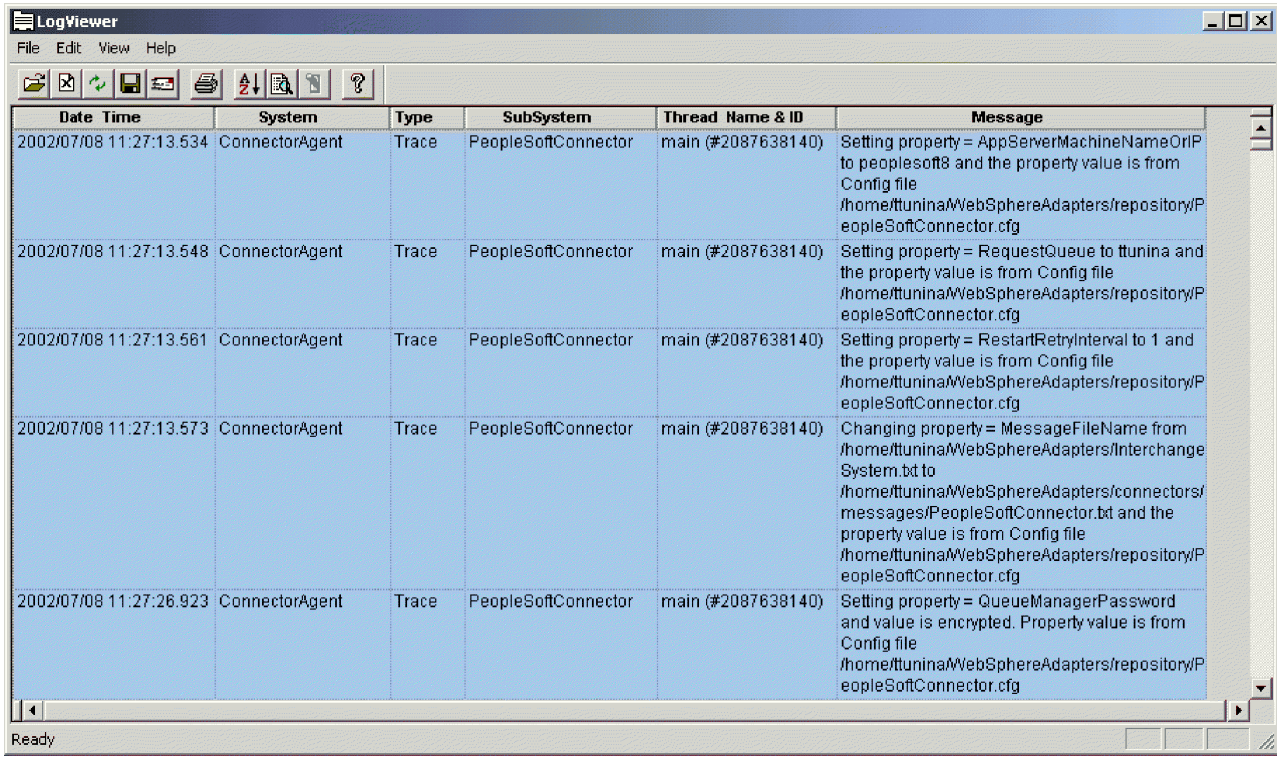


図 20. サンプル・ログを表示する LogViewer

Log Viewer の設定の指定

1. Log Viewer の設定を指定するには、メニュー・バーで**「編集」 >「設定」**と選択します。

   「ユーザー構成オプション - 一般プロパティー」ダイアログ・ボックスが表示されます (132 ページの図 21 を参照)。

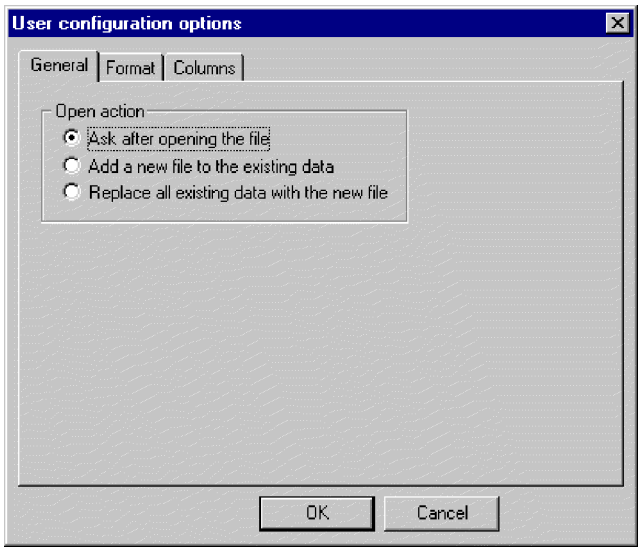


図 21. 「LogViewer ユーザー構成オプション - 一般プロパティー」画面

このダイアログ・ボックスで、ログ・ファイルを開いたときの表示方法を指定します。使用可能な選択項目は次のとおりです。

- ログ・ファイルを開くたびに設定内容を照会する。
- 開くログ・ファイルと表示中のログ・ファイルをマージする。
- 表示中のログ・ファイルを、開くログ・ファイルの内容で置き換える。

2. 背景色と Log Viewer メッセージのフォントを変更するために、**「フォーマット」**タブをクリックする。

「ユーザー構成オプション - フォーマット・プロパティー」ダイアログ・ボックスが表示されます (図 22 を参照)。

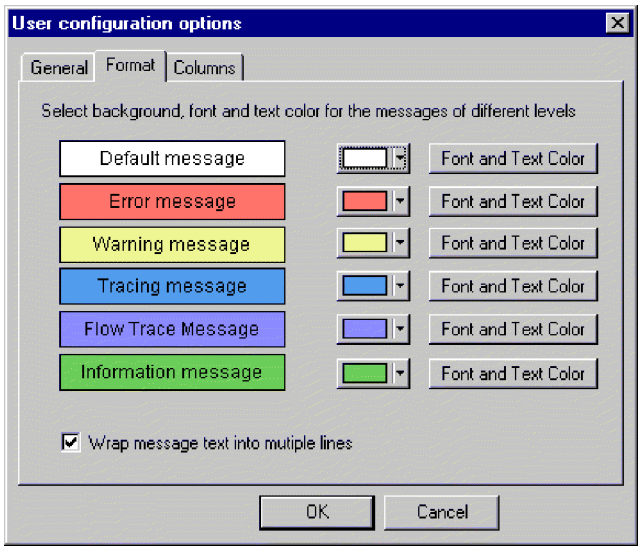


図 22. 「LogViewer ユーザー構成オプション - フォーマット・プロパティー」画面

このダイアログ・ボックスで、ログ・メッセージの表示方法を指定します。使用可能な選択項目は次のとおりです。

- 表示するメッセージ・タイプごとに異なる背景色とフォントを割り当てて、その重要度が簡単に識別できるようにする (例えば、警告メッセージの場合は赤色の背景色と大きなフォント)。
- メッセージのテキストがカラムの幅を上回る場合は、そのテキストを折り返す。

注: フロー・トレース・メッセージは、WebSphere Application Server を使用するコネクターでは生成されません。

3. 表示する Log Viewer 列を変更するには、**「列」**タブをクリックします。

「ユーザー構成オプション - 列プロパティー」ダイアログ・ボックスが表示されます (図 23 を参照)。

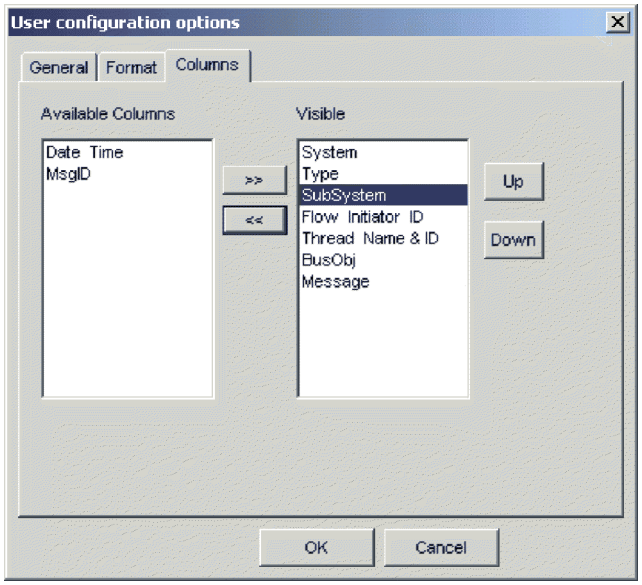


図 23. 「LogViewer ユーザー構成オプション - 列プロパティー」画面

このダイアログ・ボックスでは、Log Viewer にどの列を表示するかを、次のように指定できます。

- 列を表示するには、「使用可能な列」ペインで列名を強調表示して**「>>」**ボタンをクリックし、「可視」ペインに移動する。
- 列を非表示にするには、「可視」ペインで列名を強調表示して**「<<」**ボタンをクリックし、「使用可能な列」ペインに移動する。
- 「使用可能な列」ペイン内の任意の列名をクリックし、**「上へ」**または**「下へ」**ボタンをクリックして、Log Viewer 表示内での左から右への表示順を変更する。「上へ」ボタンで列が左へ移動し、「下へ」ボタンで列が右へ移動します。
- **「空の列を自動的に非表示にする」**の横のチェック・ボックスをクリックして、Log Viewer の表示をコンパクトにする。

注: 「フロー・イニシエーター ID」列は、WebSphere Application Server を使用するコネクターには関係ありません。

メッセージの表示法の変更

「表示」メニューには Log Viewer の表示を変更するその他のオプションが用意されています。そのメニューで、以下の機能が使用できます。

- Log Viewer ツールバーの表示/非表示。
- Log Viewer ステータス・バーの表示/非表示。
- ウィンドウの複数のビューへの分割。
- 時刻範囲などの「フィルター」タブのフィルター・オプションをチェックする、またはメッセージ・タイプによる、メッセージすべてのフィルター操作または表示 (135 ページの図 24 および 136 ページの表 11 を参照)。フィルター・オプションの設定は、次の手順で行います。

  1. メニュー・バーから**「表示」>「フィルター」>「フィルターの使用」**の順に選択する。「フィルターの設定」ダイアログ・ボックスが表示されます。
  2. 「フィルターの活動化」領域で、適用するフィルター・オプションを含んだタブに関連するボックスをクリックする。
  3. **「OK」**をクリックしてフィルターを使用可能にする。

     フィルター出力は、ツールバーの「フィルターの切り替え」ボタンで、オンまたはオフに切り替えることができます。

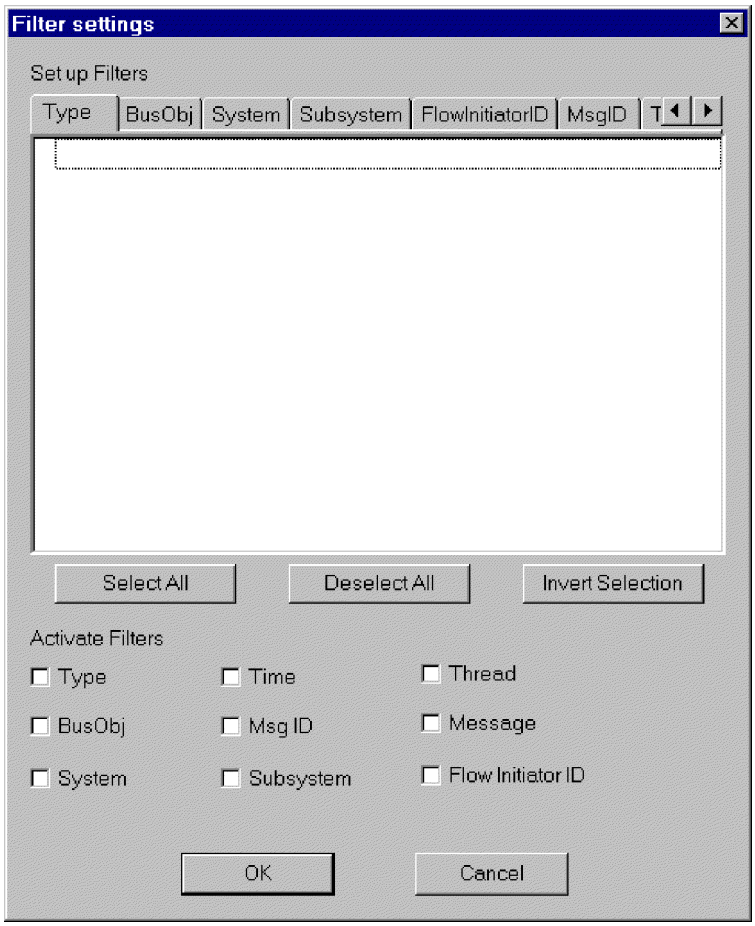


図 24. LogViewer の「フィルター」画面

注: 「フロー・イニシエーター ID」タブは、WebSphere Application Server を使用するコネクターには関係ありません。

- メッセージのソートについては、図 25 で「ソート」オプションを紹介します。各ソート・フィールドにある「下矢印」をクリックして、「日付/時刻」または「イベント ID」を選択する。昇順または降順のいずれかでソートします。

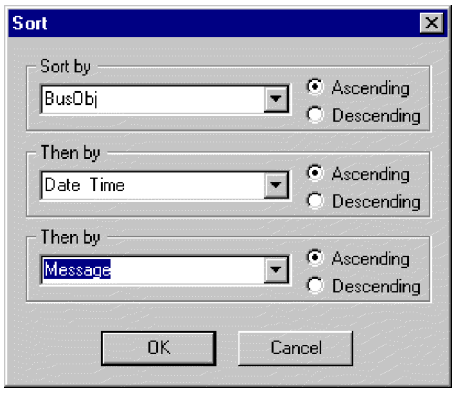


図 25. LogViewer の「ソート・プロパティー」画面

Log Viewer 表示出力の制御

Log Viewer 出力を制御するためのオプションがいくつか用意されています。 「ファイル」メニューには、印刷プレビュー、印刷、保管、最新表示、電子メール宛先への送信、さらにページ設定やヘッダーおよびフッターの、スタイルの決定に関するオプションが提供されています。ヘッダーおよびフッター用の変数は次のとおりです。

変数名	説明
$F	ファイル名。
$A	アプリケーション名。
$P	ページ番号。
$N	合計ページ数。
$D	日付 (直後にパラメーターを追加可能 (例えば、$D%y:%h:%m))。

メッセージのフィルター操作

LogViewer に表示されるメッセージをフィルター操作するには、「LogViewer」メニュー・バーから**「表示」->「フィルター」->「フィルターの使用」**の順に選択します。「フィルターの設定」ダイアログ・ボックスには、ロギング・メッセージ・フォーマットのパラメーターに対応するカテゴリーが表示されています。メッセージ・フォーマット・パラメーターを、表 11 で紹介します。

表 11. ログ・ファイルのメッセージ・フォーマット・パラメーター

変数	説明
Time	タイム・スタンプ: *年/月/日時* 形式でのロギング日付。
System	WMQI 統合ブローカーを使用するコネクターの場合、*system* 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。
Thread	スレッド名およびスレッド ID。
Name	ClarifyConnector などのコンポーネント名。
MsgType	メッセージの重大度を示す。 137 ページの表 12 を参照してください。
MsgID	メッセージ番号。
SubSystem	コネクター名。
BO	ビジネス・オブジェクト名。
MsgText	メッセージ番号の関連テキスト。
BOD	ビジネス・オブジェクト・ダンプ。ビジネス・オブジェクトに含まれているデータ。

「フィルターの設定」ダイアログで、まず使用するフィルター・カテゴリーを選択し、次に各カテゴリーから表示する項目を指定し、さらに現在の LogViewer 画面でアクティブ状態にするフィルターを選択します。

次の手順にしたがいます。

1. 「フィルターの設定」ダイアログで、「フィルターのセットアップ」にあるタブを選択して、メッセージのフィルター操作に使用する項目を表示する。例えば、

メッセージのタイム・スタンプに従ってフィルター操作する場合は、「時刻」を選択する。複数のフィルターをセットアップして、これらを別々または一緒に使用することができます。

2. 表示された項目リストから、LogViewer でメッセージを表示する項目をそれぞれ選択する。例えば、2002 年 3 月 5 日午前 9 時から 2002 年 3 月 6 日午後 5 時までのタイム・スタンプのあるメッセージのみを表示する場合、「時刻」タブでこの時間範囲を選択する。

   リスト・ボックスの下にあるボタンを使用して、表示された項目すべての選択、表示された項目すべての選択解除、または現在の選択肢の反転を行なうことができます。

3. 「フィルターの活動化」で、活動化する各フィルター・タイプのボックスを選択する。例えば、特定のタイム・スタンプで、特定のメッセージ ID を持つメッセージのみを表示する場合は、メッセージ ID フィルターと時刻フィルターの両方をアクティブ状態にする。

4. **「OK」**をクリックする。「フィルターの設定」ダイアログ・ボックスがクローズされ、LogViewer の最新表示により、フィルターを通過したメッセージのみが表示されます。

カテゴリーに従ったフィルター操作のほかに、特定のテキスト・ストリングを含むメッセージのみを表示することもできます。このためには、「フィルターのセットアップ」で「メッセージ」を選択し、表示するメッセージの特定のテキストを入力して、**「フィルターの活動化」で「メッセージ」**のボックスを選択します。

メッセージ・タイプ

表 12 で、WebSphere Business Integration Adapters から発行されるメッセージのタイプについて紹介します。

表 12. メッセージ・タイプ

メッセージ・タイプ	説明
情報	通知のみ。対処は必要ありません。
警告	InterChange Server で選択されたデフォルトの条件。
エラー	調査を必要とする重大な問題。
致命的エラー	オペレーションを停止し、報告を必要とするエラー。
トレース	指定されたトレース・レベルのトレース情報。
フロー・トレース	ビジネス・オブジェクトのフロー・トレース情報。
内部エラー	調査を必要とする、重大な内部的問題。
内部致命的エラー	オペレーションを停止する内部エラー。報告の必要あり。

注: タイプが内部エラーまたは内部致命的エラーのメッセージが表示された場合は、障害発生時での環境の状態を記録して IBM テクニカル・サポートまでご連絡ください。

付録 A. WebSphere MQ のメッセージ・フォーマット

次の表には、WebSphere MQ メッセージ・フォーマットおよび特定のプロパティーの設定値を示します。これらは、コネクター・フレームワークおよび統合ブローカーによって交換される各種のメッセージに使用されます。

表 13. コネクター・フレームワークから統合ブローカーへ送信されるイベント・デリバリー・メッセージのフォーマットおよびプロパティーの設定値

MQMD	関連情報はありません。
RFH2 メッセージ・ヘッダー	<mcd> フォルダーで、以下のフィールドには、メッセージ、そのフォーマット、およびメッセージの構文解析必要条件を識別する情報が含まれます。 • **メッセージ・ドメイン**は **mrm** に設定されます。この設定で、統合ブローカーは MRM 管理メッセージ (メッセージ・リポジトリーで完全モデルとなったメッセージ) 専用のパーサーを使用するように指定されます。 • **メッセージ・タイプ**は、メッセージが表現するトップレベルのビジネス・オブジェクトの名前を識別します。 • **メッセージ・セット**は、そのメッセージが関連付けられているメッセージ・セットを識別します。メッセージ・セットは、各タイプのビジネス・オブジェクトごとに、個別に作成されます。 • **メッセージ・フォーマット**は **CwXML** に設定されます。
メッセージ本文	RFH2 ヘッダーのメッセージ・タイプで指定されたビジネス・オブジェクトの XML スキーマとインポート済みメッセージ・セット定義に準拠した XML インスタンス文書が含まれています。

表 14. コネクター・フレームワークから統合ブローカーへ送信される要求メッセージのフォーマットおよびプロパティーの設定値

MQMD	応答先情報は、**ReplyToQ** および **ReplyToQMgr** という 2 つのフィールドに設定されています。両フィールドには、統合ブローカーがメッセージ応答メッセージを送信するのに必要な、キュー名およびキュー・マネージャー名が入っています。JMS メッセージでは、両フィールドで、要求メッセージの JMSReplyTo 宛先を指定します。MessageID (JMSMessageID) には固有値が含まれていますが、この固有値は応答メッセージ上の CorrelID プロパティー・フィールドにコピーされます。
RFH2 メッセージ・ヘッダー	<mcd> フォルダーで、以下のフィールドには、メッセージ、そのフォーマット、およびメッセージの構文解析必要条件を識別する情報が含まれます。 • **メッセージ・ドメイン**は **mrm** に設定されます。この設定で、統合ブローカーは MRM 管理メッセージ (メッセージ・リポジトリーで完全モデルとなったメッセージ) 専用のパーサーを使用するように指定されます。 • **メッセージ・タイプ**は、メッセージが表現するトップレベルのビジネス・オブジェクトの名前を識別します。 • **メッセージ・セット**は、そのメッセージが関連付けられているメッセージ・セットを識別します。メッセージ・セットは、各タイプのビジネス・オブジェクトごとに、個別に作成されます。 • **メッセージ・フォーマット**は **CwXML** に設定されます。 <jms> フォルダーで、**Rto** (JMSReplyTo) フィールドには、統合ブローカーがメッセージ応答メッセージを送信するのに必要な、キュー名およびキュー・マネージャー名をエンコードする URI が入っています。この URI を指定する方法については、「*WebSphere MQ: Java の使用*」を参照してください。MQMD の **Rto** 内および **ReplyToQ/ReplyToQMgr** 内の応答先情報は同一です。
メッセージ本文	RFH2 ヘッダーのメッセージ・タイプで指定されたビジネス・オブジェクトの XML スキーマとインポート済みメッセージ・セット定義に準拠した XML インスタンス文書が含まれています。

表 15. 統合ブローカーからコネクター・フレームワークへの応答メッセージのフォーマットおよびプロパティーの設定値

MQMD	**CorrelID** プロパティー・フィールドには、統合ブローカーが応答する要求のメッセージ ID が含まれています。JMS メッセージの場合、このフィールドは JMSCorrelationID を定義するために使用します。
RFH2 メッセージ・ヘッダー	<mcd> フォルダーで、以下のフィールドには、メッセージ、そのフォーマット、およびメッセージの構文解析必要条件を識別する情報が含まれます。 • **メッセージ・ドメイン**は **mrm** に設定されます。この設定で、統合ブローカーは MRM 管理メッセージ (メッセージ・リポジトリーで完全モデルとなったメッセージ) 専用のパーサーを使用するように指定されます。 • **メッセージ・タイプ**は、メッセージが表現するトップレベルのビジネス・オブジェクトの名前を識別します。 • **メッセージ・セット**は、そのメッセージが関連付けられているメッセージ・セットを識別します。メッセージ・セットは、各タイプのビジネス・オブジェクトごとに、個別に作成されます。 • **メッセージ・フォーマット**は **CwXML** に設定されます。 <usr> フォルダーで、以下のフィールドには、リターン状況情報が含まれています。 • **状況**フィールドには、リターン状況情報を含んだストリングが入っています。 想定されるストリングの内容は、次のいずれかです。 **-1:** 要求したオペレーションは失敗しました。 **0:** 要求したオペレーションは正常に実行されました。 **1:** 要求したオペレーションは正常に実行されました。アプリケーションから変更済みビジネス・オブジェクトが戻されました。 • **説明**フィールド - 状況が -1 に設定されている場合、このフィールドには統合ブローカーが送信したメッセージを含んだ、拡張エラー・ストリングが入っています。
メッセージ本文	RFH2 ヘッダーのメッセージ・タイプで指定されたビジネス・オブジェクトの XML スキーマとインポート済みメッセージ・セット定義に準拠した XML インスタンス文書が含まれています。

表 16. 統合ブローカーからコネクター・フレームワークへ送信される要求メッセージのフォーマットおよびプロパティーの設定値

MQMD	応答先情報は、**ReplyToQ** および **ReplyToQMgr** という 2 つのフィールドに設定されています。両フィールドには、統合ブローカーがメッセージ応答メッセージを送信するのに必要な、キュー名およびキュー・マネージャー名が入っています。JMS メッセージでは、両フィールドで、要求メッセージの JMSReplyTo 宛先を指定します。ReplyToQ および ReplyToQMgr フィールドをブランクのままにすると、コネクター・フレームワークからの応答は返ってきません。応答が必要な場合には、メッセージでメッセージ・ヘッダーの **Rto** プロパティー・フィールドに応答先情報を指定することもできます。MessageID (JMSMessageID) には固有値が含まれていますが、この固有値は応答メッセージ上の CorrelID プロパティー・フィールドにコピーされます。
RFH2 メッセージ・ヘッダー	<mcd> フォルダーで、以下のフィールドには、メッセージ、そのフォーマット、およびメッセージの構文解析必要条件を識別する情報が含まれます。 • **メッセージ・ドメイン**は **mrm** に設定されます。この設定で、統合ブローカーは MRM 管理メッセージ (メッセージ・リポジトリーで完全モデルとなったメッセージ) 専用のパーサーを使用するように指定されます。 • **メッセージ・タイプ**は、メッセージが表現するトップレベルのビジネス・オブジェクトの名前を識別します。 • **メッセージ・セット**は、そのメッセージが関連付けられているメッセージ・セットを識別します。メッセージ・セットは、各タイプのビジネス・オブジェクトごとに、個別に作成されます。 • **メッセージ・フォーマット**は **CwXML** に設定されます。 <jms> フォルダーで、**Rto** (JMSReplyTo) プロパティー・フィールドには、コネクター・フレームワークが応答メッセージを送信するのに必要なキュー名およびキュー・マネージャー名を、オプションとして含めることができます。
メッセージ本文	RFH2 ヘッダーのメッセージ・タイプで指定されたビジネス・オブジェクトの XML スキーマとインポート済みメッセージ・セット定義に準拠した XML インスタンス文書が含まれています。

表 17. コネクター・フレームワークから統合ブローカーへ送信される要求メッセージのフォーマットおよびプロパティーの設定値

MQMD	**CorrelID** プロパティー・フィールドには、コネクター・フレームワークの応答先となる要求メッセージの ID が含まれています。
RFH2 メッセージ・ヘッダー	<mcd> フォルダーで、以下のフィールドには、メッセージ、そのフォーマット、およびメッセージの構文解析必要条件を識別する情報が含まれます。 • **メッセージ・ドメイン**は **mrm** に設定されます。この設定で、統合ブローカーは MRM 管理メッセージ (メッセージ・リポジトリーで完全モデルとなったメッセージ) 専用のパーサーを使用するように指定されます。 • **メッセージ・タイプ**は、メッセージが表現するトップレベルのビジネス・オブジェクトの名前を識別します。 • **メッセージ・セット**は、そのメッセージが関連付けられているメッセージ・セットを識別します。メッセージ・セットは、各タイプのビジネス・オブジェクトごとに、個別に作成されます。 • **メッセージ・フォーマット**は **CwXML** に設定されます。 <usr> フォルダーで、以下のフィールドには、リターン状況情報が含まれています。 • **状況**フィールドには、リターン状況表示を含んだストリングが入っています。 想定されるストリングの内容は、次のいずれかです。 **-1:** 要求したオペレーションは失敗しました。 **0:** 要求したオペレーションは正常に実行されました。 **1:** 要求したオペレーションは正常に実行されました。アプリケーションから変更済みビジネス・オブジェクトが戻されました。 • 説明プロパティー・フィールド - 状況が -1 に設定されている場合、本フィールドにはコネクター・フレームワークが送信したメッセージを含んだ、拡張エラー・ストリングが入っています。
メッセージ本文	RFH2 ヘッダーのメッセージ・タイプで指定されたビジネス・オブジェクトの XML スキーマとインポート済みメッセージ・セット定義に準拠した XML インスタンス文書が含まれています。

表 18. コネクター・フレームワークから統合ブローカーへ送信される管理メッセージのフォーマットおよびプロパティーの設定値

MQMD	関連情報はありません。
RFH2 メッセージ・ヘッダー	<mcd> フォルダーで、以下のフィールドには、メッセージ、そのフォーマット、およびメッセージの構文解析必要条件を識別する情報が含まれます。 • **メッセージ・ドメイン**を **xml** に設定すると、メッセージは汎用 XML パーサーによって構文解析されるように指定されます。
メッセージ本文	メッセージ本文の内容に関する詳細については、 163 ページの『付録 C. コネクター始動オプション』を参照してください。

表 19. 統合ブローカーからコネクター・フレームワークへ送信される管理メッセージのフォーマットおよびプロパティーの設定値

MQMD	管理メッセージが「コネクターを停止 (Stop Connector)」の場合、Format プロパティーは次のとおりに設定されます。MQC.MQFMT_STRING および Expiry (JMSExpiration) プロパティー・フィールドを 1 分とする。
メッセージ本文	メッセージ本文の内容に関する詳細については、 163 ページの『付録 C. コネクター始動オプション』を参照してください。

付録 B. コネクターの標準構成プロパティー

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

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

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

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

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

新規プロパティーと削除されたプロパティー

以下の標準プロパティーは、本リリースで追加されました。

新規プロパティー

- XMLNameSpaceFormat

削除されたプロパティー

- RestartCount

標準コネクター・プロパティーの構成

アダプター・コネクターには 2 つのタイプの構成プロパティーがあります。

- 標準構成プロパティー
- コネクター固有の構成プロパティー

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

Connector Configurator の使用

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

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

プロパティー値の設定と更新

プロパティー・フィールドのデフォルトの長さは 255 文字です。

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

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

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

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

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

標準プロパティーの要約

表 20 は、標準コネクター構成プロパティーの早見表です。標準プロパティーの依存関係は RepositoryDirectory に基づいているため、コネクターによっては使用されないプロパティーがあり、使用する統合ブローカーによってプロパティーの設定が異なる可能性があります。

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

表 20. 標準構成プロパティーの要約

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

表 20. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
DuplicateEventElimination	True または False	False	コンポーネント再始動	JMS トランスポートのみ、Container Managed Events は <NONE> でなければならない
FaultQueue		*CONNECTORNAME* /FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory または CxCommon.Messaging.jms.SonicMQFactory または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	FactoryClassName が IBM の場合は crossworlds.queue.manager を使用。FactoryClassName が Sonic の場合 localhost:2506 を使用。	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMaxNativeStackSize	スタックのサイズ (キロバイト単位)	128k	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	Repository Directory は <REMOTE>
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならない
Locale	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR **注:** これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	True または False	False	コンポーネント再始動	Repository Directory は <REMOTE> でなければならない

表 20. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならない
MessageFileName	パスまたはファイル名	InterchangeSystem.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	*CONNECTORNAME*/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は True でなければならない
OADAutoRestartAgent	True または False	False	動的	Repository Directory は <REMOTE> でなければならない
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければならない
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければならない
PollEndTime	HH:MM	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒) no (ポーリングを使用不可にする) key (コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力された場合にのみポーリングする)	10000	動的	
PollQuantity	1 から 500	1	エージェント再始動	JMS トランスポートのみ: Container Managed Events を指定
PollStartTime	HH:MM (HH は 0 から 23、MM は 0 から 59)	HH:MM	コンポーネント再始動	
RepositoryDirectory	メタデータ・リポジトリーの場所		エージェント再始動	ICS: <REMOTE> に設定する WebSphere MQ Message Brokers および WAS の場合: C:¥crossworlds¥repository に設定する

表 20. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
RequestQueue	有効な JMS キュー名	*CONNECTORNAME*/REQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
ResponseQueue	有効な JMS キュー名	*CONNECTORNAME*/RESPONSEQUEUE	コンポーネント再始動	Delivery Transport が JMS の場合: Repository Directory が <REMOTE> の場合のみ必要
RestartRetryCount	0 から 99	3	動的	
RestartRetryInterval	適切な正数 (単位: 分): 1 から 2147483547	1	動的	
RHF2MessageDomain	mrm、xml	mrm	コンポーネント再始動	Delivery Transport が JMS であり、かつ WireFormat が CwXML である
SourceQueue	有効な WebSphere MQ 名	*CONNECTORNAME*/SOURCEQUEUE	エージェント再始動	Delivery Transport が JMS であり、かつ Container Managed Events が指定されている場合のみ
SynchronousRequestQueue		*CONNECTORNAME*/ SYNCHRONOUSREQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
SynchronousRequestTimeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	Delivery Transport は JMS
SynchronousResponseQueue		*CONNECTORNAME*/ SYNCHRONOUSRESPONSEQUEUE	コンポーネント再始動	Delivery Transport は JMS
WireFormat	CwXML、CwBO	CwXML	エージェント再始動	Repository Directory が <REMOTE> でない場合は CwXML。Repository Directory が <REMOTE> であれば CwBO
WsifSynchronousRequest Timeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	WAS のみ
XMLNameSpaceFormat	short、long	short	エージェント再始動	WebSphere MQ Message Brokers および WAS のみ

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

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

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

RepositoryDirectory が <REMOTE> の場合のみ適用されます。

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搬送します。

ApplicationName

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

BrokerType

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

CharacterEncoding

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

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

デフォルトでは、ドロップ・リストには、サポートされる文字エンコードの一部のみが表示されます。サポートされるその他の値をドロップ・リストに追加するに

は、製品ディレクトリーにある ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。詳細については、Connector Configurator に関する付録を参照してください。

ConcurrentEventTriggeredFlows

RepositoryDirectory が <REMOTE> の場合のみ適用されます。

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 No value です。

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

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

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

注: ContainerManagedEvents を JMS に設定すると、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が使用できなくなります。

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

ControllerStoreAndForwardMode

RepositoryDirectory が <REMOTE> の場合のみ適用されます。

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

RepositoryDirectory が <REMOTE> の場合のみ適用されます。

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。デフォルトは、0 です。

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

コネクターがビジネス・オブジェクトを統合ブローカーに送信するために使用されるキュー。

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

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

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

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

WebSphere MQ および IDL

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

- 非同期 (ASYNC) 通信:
  WebSphere MQ を使用すると、アプリケーション固有のコンポーネントは、サーバーが利用不能である場合でも、イベントをポーリングして永続的に格納することができます。
- サーバー・サイド・パフォーマンス:
  WebSphere MQ を使用すると、サーバー・サイドのパフォーマンスが向上します。最適化モードでは、WebSphere MQ はイベントへのポインターのみをリポジトリー・データベースに格納するので、実際のイベントは WebSphere MQ キュー内に残ります。これにより、サイズが大きい可能性のあるイベントをリポジトリー・データベースに書き込む必要がありません。
- エージェント・サイド・パフォーマンス:
  WebSphere MQ を使用すると、アプリケーション固有のコンポーネント側のパフォーマンスが向上します。WebSphere MQ を使用すると、コネクターのポーリング・スレッドは、イベントを選出した後、コネクターのキューにそのイベントを入れ、次のイベントを選出します。この方法は IDL よりも高速で、IDL の場合、コネクターのポーリング・スレッドは、イベントを選出した後、ネットワーク経由でサーバー・プロセスにアクセスしてそのイベントをリポジトリー・データベースに永続的に格納してから、次のイベントを選出する必要があります。

JMS

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

このプロパティーを true に設定すると、JMS 対応コネクターによるデリバリー・キューへの重複イベントのデリバリーが防止されます。この機能を使用するには、コネクターに対し、アプリケーション固有のコード内でビジネス・オブジェクトの **ObjectEventId** 属性として一意のイベント ID が設定されている必要があります。これはコネクター開発時に設定されます。

このプロパティーは、false に設定することもできます。

注: DuplicateEventElimination を true に設定した場合は、保証付きイベント・デリバリーを使用可能にするために、MonitorQueue プロパティーも構成する必要があります。

FaultQueue

コネクターは、メッセージの処理中にエラーを検出すると、当該メッセージを状況表示および問題の記述とともに、このプロパティーで指定されたキューに移動します。

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

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

デフォルト値は 128M です。

JvmMaxNativeStackSize

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

デフォルト値は 128K です。

JvmMinHeapSize

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

デフォルト値は 1M です。

jms.FactoryClassName

JMS プロバイダー用にインスタンスを生成するための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。

デフォルトは CxCommon.Messaging.jms.IBMMQSeriesFactory です。

jms.MessageBrokerName

JMS プロバイダー用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。

デフォルトは crossworlds.queue.manager です。

jms.NumConcurrentRequests

1 つのコネクターに対して同時に送信可能な並行サービス呼び出し要求の最大数を指定します。最大数に到達すると、新規のサービス呼び出しはブロックされ、ほかの要求の処理が完了してから新規のサービス呼び出しの処理が再開されます。

デフォルト値は 10 です。

jms.Password

JMS プロバイダー用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 プロバイダー用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の順序、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、

ll	2 文字の言語コード (普通は小文字)
TT	2 文字の国コードまたは地域コード (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。サポートされるその他の値をドロップ・リストに追加するには、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

RespositoryDirectory が <REMOTE> の場合のみ適用されます。

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

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

MaxEventCapacity

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

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

MessageFileName

コネクター・メッセージ・ファイルの名前です。メッセージ・ファイルの標準の格納場所は ¥connectors¥messages です。メッセージ・ファイルを標準の格納場所以外に格納する場合は、メッセージ・ファイル名を絶対パスで指定します。

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

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

MonitorQueue

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

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

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

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

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

デフォルト値は false です。

OADMaxNumRetry

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

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

デフォルト値は 1000 です。

OADRetryTimeInterval

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

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

デフォルト値は 10 です。

PollEndTime

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

このプロパティーには、有効な値をユーザーが指定する必要があります。デフォルト値は HH:MM ですが、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。

- 各ポーリング・アクション間でのミリ秒数。
- 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。入力は小文字で行います。
- ワード no。コネクターは、ポーリングを実行しません。入力は小文字で行います。

デフォルトは 10000 です。

重要: このプロパティーの使用に関して制約事項のあるコネクターもあります。特定のコネクターが該当するかどうかを判別するには、それぞれのアダプター・ガイドのインストールと構成に関する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。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、有効な値をユーザーが指定する必要があります。デフォルト値は HH:MM ですが、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ために使用されるキュー。

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

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

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

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

ResponseQueue

DeliveryTransport が JMS の場合のみ適用され、RepositoryDirectory が <REMOTE> の場合のみ必要です。

JMS 応答キューを指定します。このキューは、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に配信します。統合ブローカーが ICS の場合、サーバーは要求を送信し、JMS 応答キューの応答メッセージを待ちます。

RestartRetryCount

コネクターが再始動を試行する回数を指定します。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

コネクターが再始動を試行する間隔 (単位: 分) 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間隔が指定されます。指定可能な値の範囲は 1 から 2147483647 です。

デフォルト値は 1 です。

RHF2MessageDomain

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

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

サンプル・ヘッダーを以下に示します。

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

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

SourceQueue

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

JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、152 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

コネクターが同期要求に対する応答を待機する時間 (単位: 分) を指定します。コネクターは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージをエラー・メッセージとともに障害キューに移動します。

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

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

WsifSynchronousRequest Timeout

WAS 統合ブローカーでのみ使用されます。

コネクターが同期要求に対する応答を待機する時間 (単位: 分) を指定します。コネクターは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージをエラー・メッセージとともに障害キューに移動します。

デフォルト値は 0 です。

XMLNameSpaceFormat

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

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

デフォルト値は short です。

付録 C. コネクター始動オプション

この後の各一覧表で、Windows または UNIX でコネクターを始動する際に指定できるコマンド行オプションについて紹介します。オプションによっては、コネクターの構成ファイル内で選択したプロパティー設定値をオーバーライドするものもあります。

注: 始動スクリプトの構造と作成方法の詳細は、「*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。

Windows

表 21 に記載されたコネクター開始オプションを使用するには、下記の項目を編集してコネクターの開始条件を変更します。

- コネクターのショートカット・プロパティー。
- コネクターの始動ファイル。通常、start_*connName*.bat。

  注: ご使用のコネクターがこの構文を使用するかどうかは、ディレクトリー *ProductDir*/connectors 内を調べると確認できます。start_*connName* という名前のスクリプト (ここで、*connName* はご使用のコネクター名) が存在する場合、このファイルを編集します。
- コネクターの始動ファイル、start_connector.bat ファイル (start_*connName*.bat始動ファイルを使用しない、C++ で記述されたコネクターの場合のみ)。

注: コネクターを始動するには、-c オプションが必要です。コネクターの始動にショートカットを使用する場合、ショートカット・プロパティーにはターゲット・フィールドでこのオプションを組み込んでおく必要があります。同様に、start_connector.bat または start_*connName*.bat を使用してコネクターを始動する場合、コマンド行にこのオプションを指定する必要があります。

表 21. Windows でのコマンド行コネクター始動オプション

オプション	説明
-c *configFile*	始動時に使用する構成ファイルの絶対パス名。当オプションは必須です。
-f *pollFrequency*	各ポーリング・アクション間での時間 pollFrequency で指定される値は以下のとおりです。 • 各ポーリング・アクション間でのミリ秒数。 • key: コネクターは、コネクターの「コマンド・プロンプト」ウィンドウに英文字 p を入力したときにのみ、ポーリングを行います。入力は小文字で行います。 • no: コネクターはポーリングを行いません。入力は小文字で行います。 デフォルトは 10000 です。

表 21. *Windows* でのコマンド行コネクター始動オプション (続き)

オプション	説明
-t	**注:** インストーラーでは、コネクター側の必要に応じて、ショートカット内に -t オプションが組み込まれて (または除外されて) います。コマンド行からコネクターを始動する場合は、-t に指定する値が、ショートカットで指定した -t の値と一致していなければなりません。コネクター・プロパティー SingleThreadAppCalls をオンの状態にします。このプロパティーにより、アプリケーション固有のコネクター・コードに対してコネクター・フレームワークが行なうすべての呼び出しに 1 つのイベント・トリガー・フローがあることが保証されます。デフォルト値は false です。このプロパティーは、出荷時の値のままにしておいてください。各コネクターには、そのアーキテクチャーに応じて、当オプションに対して最適な設定値が用意されています。
-x *connectorProps*	アプリケーション固有のコネクター・プロパティーをコネクターに受け渡します。入力する値ごとに、フォーマット prop_name=*value* を使用します。

UNIX

UNIX 環境では、connector_manager_*connName* スクリプトを実行してコネクターを始動します。このスクリプトは、汎用コネクター・マネージャー・スクリプト (*ProductDir*/bin/connector_manager) のラッパーです。汎用コネクター・マネージャー・スクリプトは、該当する start_*connName*.sh スクリプトを呼び出します。このスクリプトは、コネクターに代わって実際のコネクター管理を行います。

各 WebSphere Business Integration Adapter には、start_*connName*.sh スクリプトが含まれています。このスクリプトを変更して、表 22 にある、サポートされている始動オプションのいずれかを指定することもできます。

表 22. *start_connector.sh* スクリプトのオプション

オプション	説明
-f*poll_freq*	各ポーリング・アクション間での時間 poll_freq で指定される値は以下のとおりです。 • 各ポーリング・アクション間でのミリ秒数。 • key: コネクターは、コネクターの「コマンド・プロンプト」ウィンドウに英文字 p を入力したときにのみ、ポーリングを行います。入力は小文字で行います。 • no: コネクターはポーリングを行いません。入力は小文字で行います。 デフォルトは 10000 です。-f オプションは、connector_manager_*connector* のコマンド行呼び出しで有効です。コネクター・マネージャー・スクリプトは当オプションを関連した start_*connector*.sh スクリプトに受け渡すことができます。当オプションで、コネクターの構成ファイルで指定済みのポーリング頻度がオーバーライドされます。

表 22. *start_connector.sh* スクリプトのオプション (続き)

オプション	説明
-t*threading_type*	threading_type オプションでは、スレッド化モデルを指定します。 **注:** カスタム開発コネクターを始動する場合は、-t オプションのみを使用します。WebSphere Business Integration Adapters インストーラーを使用してインストールされるコネクターには connector_manager_*connector* 始動スクリプトがすでに存在します。この始動スクリプトは、アプリケーション側の必要に応じて、コネクターを開始する行で必須の -t オプションを指定 (または除外) します。 threading_type で使用される値は以下のとおりです。 • SINGLE_THREADED: 単一スレッドのみがアプリケーションにアクセスします。 • MAIN_SINGLE_THREADED: メイン・スレッドのみがアプリケーションにアクセスします。 • MULTI_THREADED: 複数のスレッドからアプリケーションにアクセスできます。 -t オプションは、connector_manager_*connName* のコマンド行呼び出しでは*無効* です。start_*connector*.sh スクリプトの起動時に汎用の connector_manager スクリプト内で、当オプションを指定します。

付録 D. コネクター・スクリプト生成ツールの使用

コネクター・スクリプト生成ユーティリティーは、UNIX プラットフォームで実行されるコネクター用のコネクター・スクリプトを作成または変更します。このツールを使用して以下のいずれかを実行します。

- WebSphere Business Integration Adapters インストーラーを使用せずに追加したコネクターに対応する、新規のコネクター開始スクリプトを生成する。
- 正しい構成ファイルパスを組み込むために、既存のコネクター開始スクリプトを修正する。

コネクター・スクリプト生成プログラムを実行するには、以下の手順を実行します。

1. *ProductDir*/bin ディレクトリーにナビゲートする。
2. コマンド ./ConnConfig.sh. を入力する。

   コネクター・スクリプト生成プログラム画面が、図 26 のように表示されます。

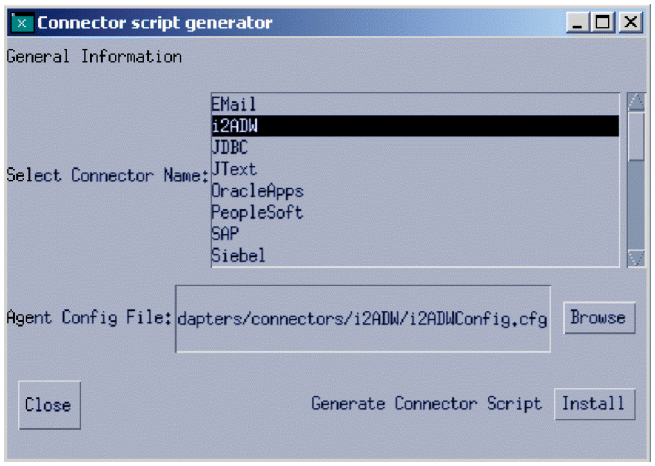


図 26. コネクター・スクリプト生成プログラム

3. 「コネクター名を選択」リストから、生成する開始スクリプトに対応するコネクターを選択する。
4. Agent Config File では、コネクターの構成ファイルの絶対パス名を入力するか、**「参照」**をクリックしてファイルを選択し、その構成ファイルを指定する。
5. コネクター・スクリプトを生成または更新するには、**「インストール」**をクリックする。

   connector_manager_*ConnectorName* ファイル (ここで、*ConnectorName* は構成するコネクターの名前) が *ProductDir*/bin ディレクトリーに作成されます。
6. **「閉じる」**をクリックする。

付録 E. System Manager と Eclipse Workbench

この付録では、System Manager と Eclipse Workbench の機能の使用について、一般的な説明の一部を示します。

注: 一部の System Manager 機能の詳細は、本書で説明されています。Adapter Monitor および障害キュー・マネージャーの詳細については、 123 ページの『Adapter Monitor および障害キュー・マネージャーの使用』を参照してください。新規プロジェクトの作成については、 75 ページの『第 8 章 WebSphere Application Server への配置』を参照してください。

この章には、以下のセクションが含まれています。

- 『System Manager とは』
- 172 ページの『System Manager の使用』
- 176 ページの『統合コンポーネント・ライブラリーでの作業』
- 178 ページの『ユーザー・プロジェクトでの作業』
- 183 ページの『統合コンポーネント・ライブラリー内のコンポーネントの処理』
- 187 ページの『ソリューションの処理』
- 190 ページの『System Manager を使用したパッケージへのコンポーネントのエクスポート』
- 192 ページの『依存関係および参照』
- 193 ページの『複数のワークベンチ・リソースで使用可能な標準の操作』
- 195 ページの『Eclipse ベースのワークベンチの使用』
- 201 ページの『System Manager で統合ブローカーに接続する際の問題のトラブルシューティング』

System Manager とは

System Manager とは、WebSphere Studio Workbench および WebSphere Studio Application Developer Integration Edition という Eclipse ベースのツール・フレームワーク内で稼動するプラグインです。このセクションでは、Eclipse フレームワーク (WSWB と WSADIE) および IBM WebSphere アダプター・ツールのプラグインの概要について説明します。

Eclipse プラットフォームとは

Eclipse プラットフォームとは、ツール作成用のオープン・ソースの統合開発環境 (IDE) です。これにより、ツール開発者に開発キットとランタイムが提供され、開発者は、ユーザーが特定のタイプのリソースを使用して作業できるようなプラグインを作成することができます。

IBM では、WebSphere Studio WorkBench (WSWB) と WebSphere Studio Application Developer Integration Edition (WSADIE) という 2 つのバージョンの Eclipse プラットフォームを用意しています。

プラグイン

プラグインとは、ソフトウェア・ベンダーが開発して Eclipse ベースのワークベンチに機能性を追加するモジュラー拡張機能です。プラグインには、ワークベンチのユーザーが特定のタイプのリソースを使用して作業するためのパースペクティブ、エディター、およびビューがカプセル化されます。

例えば、あるプラグインはテキスト・エディター機能を提供します。別のプラグインは、HTML エディター機能を提供します。 System Manager ツールのプラグインは、統合コンポーネントと連動する機能を提供します。プラグイン・モデルの利点は、ユーザーが多様なリソースを使用して作業する際に、各リソース・タイプの専用ツールではなく、単一のツールを使用できることです。

プラグインをインストールするには、プラグインを示す 1 つ以上の圧縮アーカイブを、ワークベンチの製品ディレクトリー内の `plugins` ディレクトリーに解凍します。System Manager プラグインが、インストーラーによって `plugins` ディレクトリーに解凍されます。

IBM では、ビジネス・インテグレーション・リソースを使用して作業するために、System Manager プラグインを WebSphere Business Integration Adapter フレームワークとともに提供します。これらのプラグインは、ワークベンチの `plugins` ディレクトリー内の多数の解凍ディレクトリーによって `com.ibm.btools` ネーム・スペース内に統合されます。統合コンポーネントの作成時に使用する基本インターフェースのほとんど (例えば System Manager) は、`com.ibm.btools.csm` プラグイン・ディレクトリーに含まれます。

ワークベンチ

ワークベンチとは、Eclipse ベースのツール・フレームワーク内でアクティブなパースペクティブ、エディター、およびビューの集合であり、すでにインストールされて使用可能なプラグインの集合からの影響を受けます。ワークベンチは、ユーザーが作業している Eclipse ベースのインターフェースを指す一般用語であり、そのインターフェースがユーザーの使用に応じて変更されることとは関係ありません。

ワークスペース

ワークスペースとは、プロジェクトのコンテナーです。ワークスペースは、デフォルトでプロジェクトの保管先にするようプロンプトが出される、ファイル・システム内のディレクトリーです。

プロジェクト

プロジェクトとは、ユーザー定義のリソース・グループであり、最終的にはファイル・システム内のディレクトリーです。

ビジネス・プロセス・インターフェースを開発する際の最初の作業の 1 つは、統合コンポーネント・ライブラリー、つまり開発するコンポーネントを含むプロジェクトを定義することです。統合コンポーネント・ライブラリーを作成するときは、その保管先となるファイル・システム内の場所を指定します (デフォルトはワークスペース・ディレクトリー)。その場所に、統合コンポーネント・ライブラリーに指定した名前をもつフォルダーが作成されます。ライブラリー・フォルダー内には、統合コンポーネントのタイプごとに多数のフォルダーが作成されます (例えば、`BusinessObjects` や `Connectors` という名前のフォルダーが作成されます)。

また、**ユーザー・プロジェクト**という名前のプロジェクトも作成します。ユーザー・プロジェクトとは、統合コンポーネントを参照するショートカットの集合です。コンポーネントを統合ブローカーに配置するには、統合コンポーネント・ライブラリーの統合コンポーネントをユーザー・プロジェクトに追加する必要があります。ユーザー・プロジェクトは、コンポーネントをサーバーに配置するのに必要であるほか、ユーザーがコンポーネントを機能的にグループ化できるよう設計されています。統合コンポーネント・ライブラリーは、作業に必要なすべてのコンポーネントの集合ですが、ユーザー・プロジェクトは、ユーザーが作業するコンポーネントを特定のインターフェース用にグループ化できるよう設計されています。

リソース

リソースとは、ユーザーがワークベンチ内で作業するプロジェクト、ファイル、およびフォルダーです。

統合コンポーネントを作成すると、統合コンポーネント・ライブラリー・プロジェクト内の該当するフォルダーにファイルとして保管されます。統合コンポーネントは、タイプごとに異なる拡張子が付けられて保管されます (例えば、コネクターには .con という拡張子が付けられます) が、すべて XML 形式で保管されます。

パースペクティブ

パースペクティブとはエディターやビューをグループ化したものであり、特定ユーザーの役割に必要なものを提供します。例えば、System Manager パースペクティブは、統合コンポーネント・ライブラリーでの作業用のビューを提供します。

エディター

エディターを使用すると、ワークベンチでリソースをオープン、保管、およびクローズできます。

ビュー

ビューは、ユーザーがワークベンチで作業しているリソースについての情報を提供します。

例えば System Manager には、統合コンポーネント・ライブラリーおよびユーザー・プロジェクトに対するビューとして、WebSphere Business Integration System Manager ビューがあります。

WSWB および WSADIE とは

WebSphere Studio Workbench (WSWB) は、IBM リリースの Eclipse プラットフォームです。WSWB は統合ブローカーに組み込まれているので、コア・インフラストラクチャーとともにインストールすることができます。 WSWB は、WebSphere Business Integration Adapters 統合コンポーネントの作成に必要なすべてのプラグインを実行することができます。

WebSphere Studio Application Developer Integration Edition (WSADIE) は、WSWB と同様に IBM リリースの Eclipse プラットフォームですが、新規プラグインの開発に使用することもできます。WSADIE は統合ブローカーに組み込まれていません。新規プラグインの作成機能は、統合コンポーネントの作成には必要ないためです。ただし、これをインストールすると、必要な System Manager プラグインおよび Integrated Test Environment プラグインを実行する際に使用できます。

System Manager とは

System Manager とは、ユーザーが WebSphere Business Integration Adapters ビジネス・インテグレーション・システムの統合コンポーネントおよびサーバー・インスタンスの作業を行うパースペクティブです。System Manager は、主に次の作業に使用します。

- WebSphere Business Integration Adapters ツール・セット内で他のツールを起動する
- 一部の統合コンポーネントを開発および構成する
- 統合コンポーネントをリポジトリーまたはブローカーに配置する

System Manager の使用

このセクションでは、System Manager パースペクティブの始動方法と使用方法について説明します。

System Manager の始動

IBM WebSphere 統合ブローカーをインストールする場合には、ブローカーとの対話に必要なプラグインのサポートとともに WebSphere Studio Workbench をインストールするか、既存の WebSphere Studio Application Developer Integration Edition にプラグインをインストールするかを選択できます。

System Manager を始動するには、次の手順を実行します。

1. **「スタート」 > 「プログラム」 > 「WebSphere Business Integration Adapters」 > 「ツール」 > 「System Manager」**を選択します。
2. メニュー・バーから、**「Windows」 > 「パースペクティブを開く」 > 「その他」**を選択します。
3. パースペクティブのリストから `System Manager` を選択し、**「OK」**をクリックします。

   WebSphere Studio Workbench が始動し、表示されます。 173 ページの図 27 に、System Manager パースペクティブを示します。『System Manager インターフェース』では、インターフェースおよびそのエレメントについて説明します。

System Manager インターフェース

System Manager パースペクティブが開くときのデフォルト構成には、幾つかのビューとエディターがあります。 173 ページの図 27 に、System Manager パースペクティブを示します。

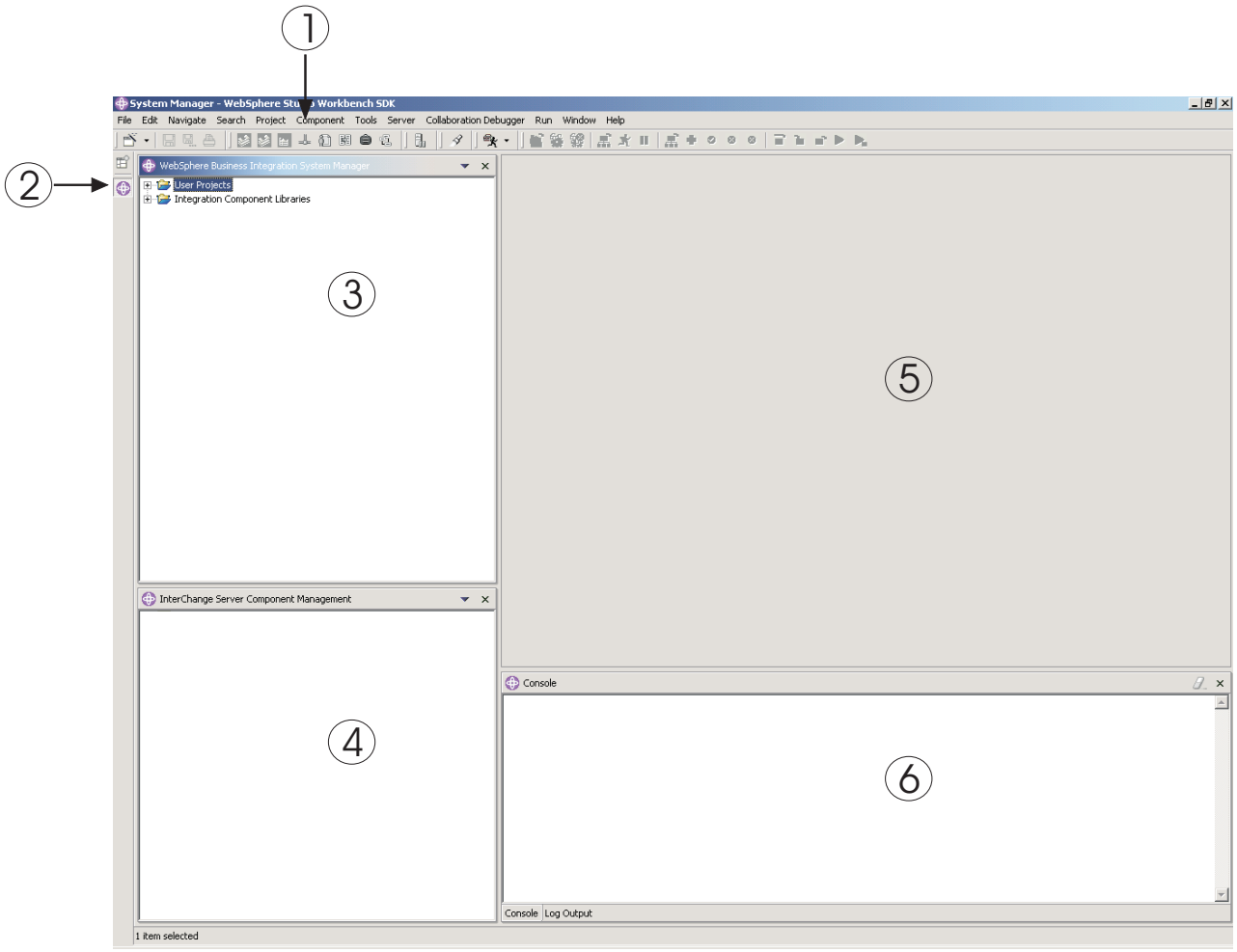


図 27. System Manager パースペクティブ

表 23 では、System Manager パースペクティブのインターフェース・エレメントについて、173 ページの図 27 の番号ごとに説明します。

表 23. System Manager パースペクティブのインターフェース・エレメント

インターフェース・エレメント番号	インターフェース・エレメント名
1	174 ページの『メニュー・バーとツールバー』
2	175 ページの『パースペクティブ・ショートカット・バー』
3	175 ページの『WebSphere Business Integration System Manager ビュー』
4	Interchange Server Component Management ビュー。このセクションが関連するのは、WebSphere Interchange Server を統合ブローカーとして実行している場合のみです。別のブローカーを使用している場合は、無視して構いません。
5	175 ページの『エディター・ビュー』
6	176 ページの『コンソール・ビュー』

以降のセクションでは、System Manager のインターフェース・エレメントについて詳しく説明します。

メニュー・バーとツールバー

メニュー・バーとツールバーは、Eclipse ベースのツール・フレームワークおよび統合ブローカー・コンポーネントでの作業を行う場合に使用します。メニュー・バー項目の多くにはツールバーと同等のものがあるので、以降のセクションではメニュー・バーとその項目のみを説明します。

「ファイル」メニュー: リソースの作業に使用する Eclipse 標準メニュー。主に、新規の統合コンポーネント・ライブラリーおよびユーザー・プロジェクトの作成に使用します。

統合コンポーネント・ライブラリーおよびユーザー・プロジェクトの作成の詳細については、176 ページの『統合コンポーネント・ライブラリーでの作業』および 178 ページの『ユーザー・プロジェクトでの作業』を参照してください。

「編集」メニュー: 「切り取り」、「コピー」、「貼り付け」など多数の標準項目がある Eclipse 標準メニュー。

作成したコンポーネントの切り取り、コピー、貼り付け、および削除の詳細については、183 ページの『統合コンポーネント・ライブラリー内のコンポーネントの処理』を参照してください。

「ナビゲート」メニュー: ワークベンチ内のリソース間をナビゲートするための Eclipse 標準メニュー。このメニューの詳細については、ワークベンチの資料を参照してください。

「検索」メニュー: リソースの検索およびリソース内の検索を可能にする Eclipse 標準メニュー。このメニューの詳細については、ワークベンチの資料を参照してください。

「プロジェクト」メニュー: プロジェクト・リソースを操作するためのメニュー項目を持つ Eclipse 標準メニュー。このメニュー項目は、System Manager パースペクティブの作業を行う際には使用されません。このメニューの詳細については、ワークベンチの資料を参照してください。

「コンポーネント」メニュー: このメニューは System Manager パースペクティブとともに提供され、ユーザーが作成した統合コンポーネントの作業を行う際に便利です。このメニューの項目は、本書内および他の資料の特定のタスクについて説明しているセクションに文書化されています。

「ツール」メニュー: このメニューは System Manager パースペクティブとともに提供され、統合コンポーネントの作成に使用するツールの起動に使用されます。

詳しくは、183 ページの『統合コンポーネント・ライブラリー内のコンポーネントの処理』を参照してください。

「サーバー」メニュー: このメニューの**「サーバーの登録 (Register Servers)」**項目を使用すると、InterChange Server インスタンスを登録できます。統合ブローカーが InterChange Server 以外の場合は、この項目は関係ありません。

「実行」メニュー: このメニューには、外部プログラムやバッチ・ファイルを実行するための外部ツールを構成したり、スクリプトを作成するための項目があります。詳しくは、ワークベンチの資料を参照してください。

「ウィンドウ」メニュー: このメニューには、パースペクティブ、ビュー、エディター、および設定での作業を行うための項目があります。

これらの項目については、ワークベンチの資料および本書のさまざまなセクションで説明しています。

「ヘルプ」メニュー: このメニューには、ワークベンチの資料を起動したり、ワークベンチおよびパースペクティブのバージョン情報を示す項目があります。

パースペクティブ・ショートカット・バー

パースペクティブ・ショートカット・バーを使用すると、異なるパースペクティブ間を簡単にナビゲートできます。例えば、System Manager パースペクティブと Java パースペクティブを開いている場合、パースペクティブ・ショートカット・ツールバーのワークスペース・アイコンをクリックすると、パースペクティブを切り替えることができます。

また、次のように**「ウィンドウ」**メニューを使用して、他のパースペクティブにナビゲートすることもできます。

- パースペクティブ・ショートカット・バーにおいて、現在表示しているパースペクティブのアイコンよりも上にあるアイコンのパースペクティブにナビゲートするには、キーボード・ショートカット **Alt + 上矢印** を使用します。
- パースペクティブ・ショートカット・バーにおいて、現在表示しているパースペクティブのアイコンよりも下にあるアイコンのパースペクティブにナビゲートするには、メニュー・バーから**「パースペクティブ (Perspective)」 > 「次へ」**を選択するか、キーボード・ショートカット **Alt + 下矢印** を使用します。

WebSphere Business Integration System Manager ビュー

このビューには、統合ブローカー・プロジェクトのタイプである**ユーザー・プロジェクト**・ノードと**統合コンポーネント・ライブラリー**・ノードがあります。

これらのプロジェクト・タイプの作業の詳細については、176 ページの『統合コンポーネント・ライブラリーでの作業』および 178 ページの『ユーザー・プロジェクトでの作業』を参照してください。

InterChange Server Component Management ビュー

このビューは、統合ブローカーとして WebSphere InterChange Server を使用している場合に使用できます。その他の統合ブローカーの場合は関係ありません。

エディター・ビュー

このビューは、フレームワーク内でさまざまなリソース (ファイルや統合コンポーネント定義など) の作業を行う場合に使用します。異なるタイプのリソースの作業を行うには、それぞれ別のエディターを開きます。例えば、テキスト・ファイルはテキスト・エディター内で開きます。

コンソール・ビュー

このビューには、**「コンソール」**と**「ログ出力」**という 2 つのタブがあります。System Manager でマップまたはコラボレーション・テンプレートをコンパイルする場合、**「コンソール」**タブには、各コンポーネントのコンパイルが正常に完了したかどうかを示すメッセージが表示され、**「ログ出力」**タブには、発生したエラーや警告が表示されます。

統合コンポーネント・ライブラリーでの作業

統合コンポーネント・ライブラリーは、作成したコンポーネントを保管するために使用します。このセクションでは、新規統合コンポーネント・ライブラリーの作成方法について説明します。

統合コンポーネント・ライブラリーを作成すると、一般的に次の作業も実行するようになります。

- ライブラリーにコンポーネントをインポートする。この作業を行うための方法については、183 ページの『統合コンポーネント・ライブラリー内のコンポーネントの処理』を参照してください。
- ユーザー・プロジェクト内にコンポーネントへのショートカットを作成する。この作業の詳細については、178 ページの『ユーザー・プロジェクトでの作業』を参照してください。
- 統合ブローカーにコンポーネントを配置する。詳細については、75 ページの『第 8 章 WebSphere Application Server への配置』を参照してください。
- コンポーネントをサーバーや他のライブラリーにインポートしたり、開発をバックアップするために、コンポーネントをパッケージにエクスポートする。詳細については、190 ページの『System Manager を使用したパッケージへのコンポーネントのエクスポート』を参照してください。

統合コンポーネント・ライブラリーの作成

ウィザードを使用して System Manager 内に新規統合コンポーネント・ライブラリーを作成するには、次の手順を実行します。

1. 次のいずれかを実行して、「新規統合コンポーネント・ライブラリー」ウィザードを始動します。
   - メニュー・バーから、**「ファイル」 > 「新規」 > 「統合コンポーネント・ライブラリー」**を選択します。
   - WebSphere Business Integration System Manager ビューで、**統合コンポーネント・ライブラリー**・フォルダーを右マウス・ボタン・クリックし、コンテキスト・メニューから**「新規統合コンポーネント・ライブラリー」**を選択します。
   - ツールバーの**「新規ウィザードを開く (Open The New Wizard)」**ボタンをクリックし、メニューから**「新規統合コンポーネント・ライブラリー」**を選択します。

   177 ページの図 28 に、「新規統合コンポーネント・ライブラリー」ウィザードを示します。

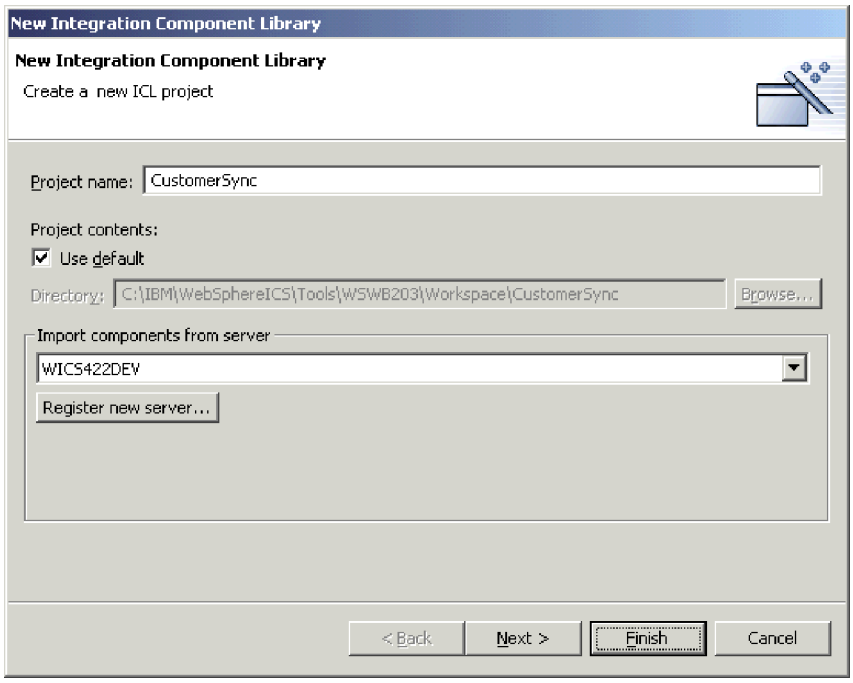


図 28. 統合コンポーネント・ライブラリーの作成

2. **「プロジェクト名」**フィールドに、統合コンポーネント・ライブラリーの名前を入力します。

   プロジェクト名には英数字および下線のみが使用でき、英語で指定する必要があります。

   ライブラリーに名前を付ける際には、ライブラリーを対応するサーバーに関連付けるようにすることをお勧めします。例えば、サーバー名が `SERV420DEV` である場合は、ライブラリーに `SERV420DEVICL` と名前を付けます。
3. ライブラリーのフォルダーをデフォルトの場所 (ワークスペース) に作成し、ライブラリーに指定した名前と同じ名前を付ける場合は、**「デフォルトを使用」**チェック・ボックスをチェックしたままにします。

   ライブラリー・フォルダーの名前と場所を指定する場合は、次の手順を実行します。

   a. **「デフォルトを使用」**チェック・ボックスをクリアします。
   b. **「ディレクトリー」**フィールドに、ライブラリー用に使用するディレクトリーの絶対パスと名前を入力するか、**「参照」**をクリックして既存のディレクトリーを選択します。

   注: ワークスペースのパスにライブラリー用のフォルダーを作成するには、この**「デフォルトを使用」**チェック・ボックスを使用して System Manager に作成させる以外に方法はありません。
4. **「終了」**をクリックして、ウィザードを終了します。

   System Manager は、**統合コンポーネント・ライブラリー・**フォルダーの下にユーザーが指定した名前のフォルダーを作成します。

ユーザー・プロジェクトでの作業

ユーザー・プロジェクトには、1 つ以上のライブラリー内で作業を行う統合コンポーネントへのショートカットを作成します。ユーザー・プロジェクトを使用すると、コンポーネントのビューをインターフェースとして編成できます。System Manager からサーバーにコンポーネントを配置するには、ユーザー・プロジェクトにコンポーネントのショートカットを追加する必要があります。

このセクションの内容は次のとおりです。

- 『ユーザー・プロジェクト用に統合ブローカー設定を構成する』
- 『ユーザー・プロジェクトの作成』
- 180 ページの『ユーザー・プロジェクトへのショートカットの追加』
- 187 ページの『ソリューションのエクスポート』

ユーザー・プロジェクト用に統合ブローカー設定を構成する

System Manager を使用して、複数の統合ブローカー用にユーザー・プロジェクトを作成することができます。さまざまな統合ブローカーに対して System Manager を使用可能にするには、次の手順を実行します。

1. テキスト・エディターで、*ProductDir*¥bin の cwtools.cfg というファイルを開きます。
2. System Manager を WebSphere Application Server プロジェクト用に使用可能にするには、WAS_PROJECT セクションの Installed プロパティーの値を true に設定します。

   System Manager を WebSphere MQ Integrator Broker プロジェクト用に使用可能にするには、WMQI_PROJECT セクションの Installed プロパティーの値を true に設定します。 System Manager を WebSphere InterChange Server プロジェクト用に使用可能にするには、ICS_PROJECT セクションの Installed プロパティーの値を true に設定します。
3. ファイルを保管して閉じます。

ユーザー・プロジェクトの作成

注: 以下は、ユーザー・プロジェクトの作成方法の一般的な説明です。新規 WebSphere Application Server プロジェクトの作成に関する具体的な説明については、75 ページの『第 8 章 WebSphere Application Server への配置』を参照してください。

System Manager でウィザードを使用して新規ユーザー・プロジェクトを作成するには、以下の手順を実行します。

1. 以下のいずれかを実行し、「新規ユーザー・プロジェクト」ウィザードを開始します。
   - メニュー・バーから**「ファイル」>「新規」>「ユーザー・プロジェクト」**を選択する。

- WebSphere Business Integration System Manager ビューで、**「ユーザー・プロジェクト」**フォルダーを右マウス・ボタンでクリックして、コンテキスト・メニューから**「新規ユーザー・プロジェクト」**を選択し、**「WAS プロジェクト」**を選択する。
- WebSphere Business Integration System Manager ビューで、**「ユーザー・プロジェクト」**フォルダーを展開し、**「WAS プロジェクト」**フォルダーを右マウス・ボタンでクリックして、コンテキスト・メニューから**「新規 WAS プロジェクト」**を選択する。
- ツールバーの**「新規ウィザードを開く (Open The New Wizard)」ボタン**をクリックし、メニューから**「新規ユーザー・プロジェクト」**を選択する。

2. **「プロジェクト名」**フィールドにユーザー・プロジェクト名を入力します。

   プロジェクト名には英数字および下線のみが使用でき、英語で指定する必要があります。

   このタイプのユーザー・プロジェクトには、対応するサーバーに関連する方法で名前を付けることをお勧めします。例えば、サーバー名が SERV420DEV の場合は、ライブラリーに SERV420DEVUP という名前を付けます。

3. デフォルトのロケーション (ワークスペース) にユーザー・プロジェクト用フォルダーを作成し、ユーザー・プロジェクトに指定した名前と同じ名前を付けるには、「プロジェクト内容」ペインの**「デフォルトの使用」**チェック・ボックスを有効にしておきます。

   ユーザー・プロジェクト・フォルダーの名前およびロケーションを指定する場合は、以下の手順を実行します。

   a. 「プロジェクト内容」ペインの**「デフォルトの使用」**チェック・ボックスをクリアします。
   b. ユーザー・プロジェクトに使用するディレクトリーの絶対パスおよび名前を**「ディレクトリー」**フィールドに入力するか、**「ブラウズ」**をクリックして既存のディレクトリーを選択します。

   注: **「デフォルトを使用」**チェック・ボックスを使用して System Manager で作成する以外に、ワークスペースのパスにユーザー・プロジェクト用フォルダーを作成する方法はありません。

4. この時点で既存の統合コンポーネントへのショートカットを作成しない場合は、ステップ 5 (180 ページ) に進みます。

   既存の統合コンポーネントへのショートカットを作成する場合は、統合コンポーネント・ライブラリーの横にあるチェック・ボックスを有効にしてその中にあるすべてのコンポーネントへのショートカットを作成するか、統合コンポーネント・ライブラリー・フォルダーを展開してコンポーネント・グループの横にあるチェック・ボックスを有効にするか、グループのフォルダーを展開して個々のコンポーネントのチェック・ボックスを有効にします。

   注: 複数の統合コンポーネント・ライブラリーから同じ名前のコンポーネントを選択する場合は、選択内容に重複する参照が存在することを通知するプロンプトは出されません。重複するコンポーネントを選択した場合は、ウィザー

ドで選択したときにライブラリーの最下部にあった統合コンポーネント・ライブラリーのコンポーネントに対してショートカットが作成されます。

図 29 に、「新規ユーザー・プロジェクト」ウィザードを示します。

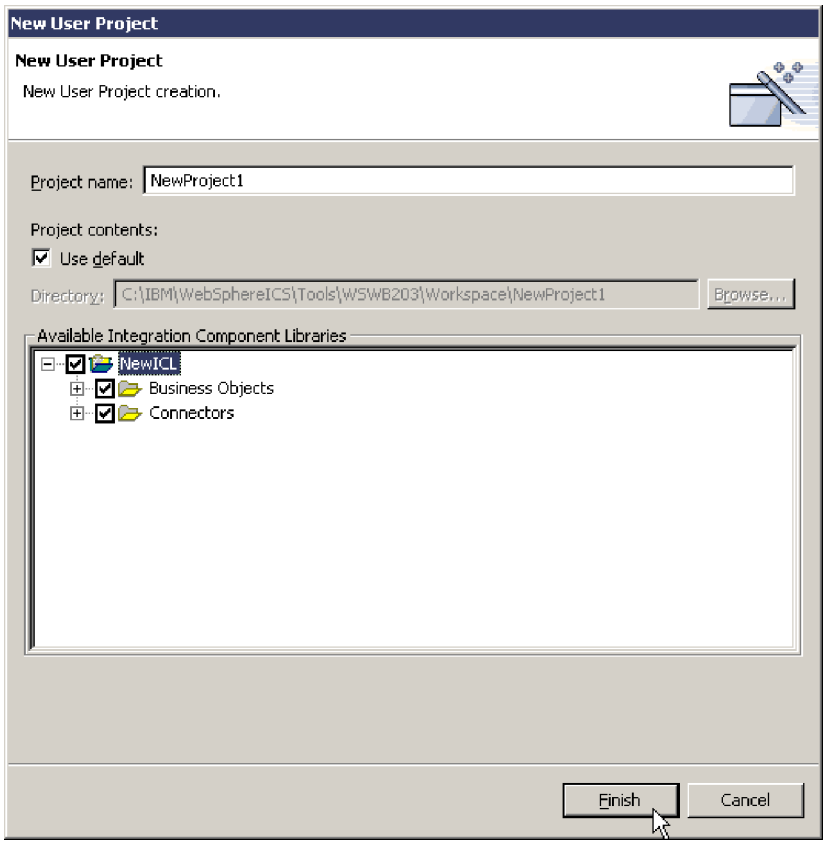


図 29. ユーザー・プロジェクトの作成

5. **「終了」**をクリックして、ウィザードを終了します。

System Manager は、**「ユーザー・プロジェクト」**フォルダーに、統合ブローカー (WAS プロジェクト) に対応するフォルダー内に指定した名前でフォルダーを作成します。

ユーザー・プロジェクトへのショートカットの追加

作業しているインターフェースを表示できるようにするには、ユーザー・プロジェクトにショートカットを追加します。ユーザー・プロジェクトにショートカットを追加する方法はいくつかあります。以下のセクションではこれらの方法について説明します。

- 181 ページの『依存関係ツリーの使用』
- 182 ページの『「プロジェクトを更新」ウィザードの使用』
- 183 ページの『コンポーネントのドラッグ・アンド・ドロップ』
- 189 ページの『ソリューションのインポート』

- 189 ページの『ソリューションのインポート』

依存関係ツリーの使用

「依存関係ツリー」ウィザードは、コンポーネント・ショートカットをユーザー・プロジェクトに追加するための最も便利なインターフェースです。ユーザー・プロジェクトは、主にインターフェースを表す目的で設計されており、インターフェースは一般にコネクター・オブジェクトを中心に置きます。さらに、ユーザー・プロジェクトは一般にコネクター・オブジェクトを中心に置きます。通常、ユーザー・プロジェクトに必要なショートカットを作成するには、コネクター・オブジェクトの依存関係を見つけます。

依存関係の概念については、 192 ページの『依存関係および参照』を参照してください。

「依存関係ツリー」ウィザードを使用してユーザー・プロジェクトにショートカットを追加するには、以下の手順を実行します。

1. ライブラリーにあるコネクター・オブジェクトなどの統合コンポーネントを右マウス・ボタンでクリックし、コンテキスト・メニューから**「依存関係を表示」**を選択します。
2. **「プロジェクトに追加」**ドロップダウン・メニューから、ショートカットの追加先ユーザー・プロジェクトを選択します。
3. ウィザードの左側のペインで、ショートカットを作成するコンポーネントを選択します。

   キーボード・ショートカットを使用すると作業しやすくなります。例えば、オブジェクトの範囲を選択するには **Shift** を押しながら操作し、不連続な単一オブジェクトを選択するには **Ctrl** を押しながら操作します。
4. 右向きの矢印をクリックし、ウィザードの右側のペインにコンポーネントを追加します。

   182 ページの図 30 に、「依存関係ツリー」ウィザードを示します。

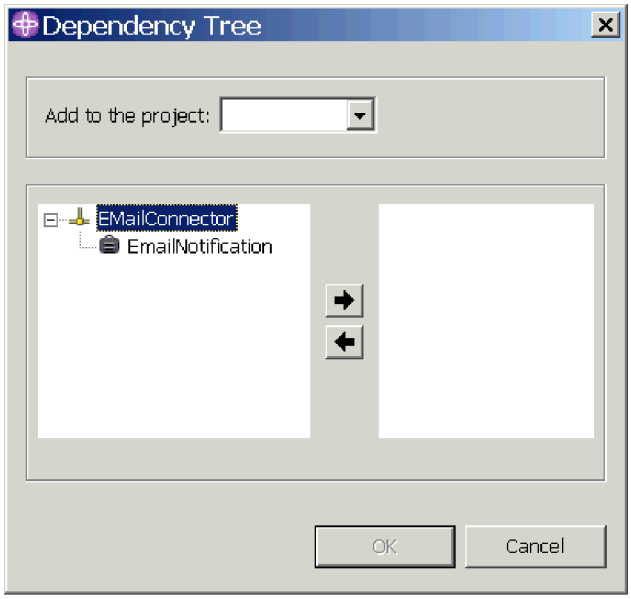


図 30. 「依存関係ツリー」を使用したユーザー・プロジェクトへのショートカットの追加

5. **「OK」**をクリックします。

   System Manager は、指定のユーザー・プロジェクトに選択されたコンポーネントへのショートカットを作成します。

「プロジェクトを更新」ウィザードの使用

「プロジェクトを更新」ウィザードが提供するインターフェースにより、最初にユーザー・プロジェクトを作成するときに表示されたユーザー・プロジェクトに類似したユーザー・プロジェクトへのショートカットを追加できます。「プロジェクトを更新」ウィザードを使用するには、以下の手順を実行します。

1. WebSphere Business Integration System Manager ビューで任意のユーザー・プロジェクトを右マウス・ボタンでクリックし、コンテキスト・メニューから**「プロジェクトを更新」**を選択します。
2. 統合コンポーネント・ライブラリーの横にあるチェック・ボックスを有効にしてその中にあるすべてのコンポーネントへのショートカットを作成するか、統合コンポーネント・ライブラリー・フォルダーを展開してコンポーネント・グループの横にあるチェック・ボックスを有効にするか、グループのフォルダーを展開して個々のコンポーネントのチェック・ボックスを有効にします。
3. **「終了」**をクリックします。

   ウィザードで選択したコンポーネントと同じ名前のコンポーネントへのショートカットがユーザー・プロジェクトに存在する場合は、プロンプトが表示され、以下の操作が可能になります。

   - 表示されたコンポーネントを上書きする。
   - 重複するコンポーネントをすべて上書きする。
   - コンポーネントを上書きしない。
   - 更新操作をキャンセルする。

ユーザー・プロジェクトにショートカットが存在するコンポーネントと同じ名前のコンポーネントを選択しなかった場合は、プロジェクトにショートカットが追加され、ウィザードが終了します。

コンポーネントのドラッグ・アンド・ドロップ

統合コンポーネント・ライブラリー・フォルダーからコンポーネントを選択し、ユーザー・プロジェクトにドラッグ・アンド・ドロップすると、そのユーザー・プロジェクトにコンポーネントへのショートカットを追加できます。

ユーザー・プロジェクトのフォルダーにまだショートカットが存在しない場合は、フォルダー自体にコンポーネントをドラッグ・アンド・ドロップしてください。フォルダーにコンポーネントをドラッグ・アンド・ドロップし、マウス・ポインターの下に四角形が表示されたらマウス・ボタンを放します。

ユーザー・プロジェクトのフォルダーにすでにショートカットが存在する場合は、フォルダー自体にコンポーネントをドラッグ・アンド・ドロップすることはできません。線が表示されるまでフォルダーの既存のショートカットの間にコンポーネントをドラッグ・アンド・ドロップし、マウス・ボタンを放します。

統合コンポーネント・ライブラリー内のコンポーネントの処理

WebSphere Business Integration システムを実装する場合、大半の時間が統合コンポーネントの処理に費やされます。本書では個々のコンポーネントの作成方法の詳細については説明しません。このセクションでは、Designer ツールの起動方法、新規コンポーネントの作成方法、既存のコンポーネントの変更方法、および System Manager で作成する一部のコンポーネントの処理方法について説明します。

統合コンポーネントの作成方法について詳しくは、以下のマニュアルを参照してください。

- *ビジネス・オブジェクト開発ガイド*
- *「コネクター開発ガイド (Java 用)」または「コネクター開発ガイド (C++ 用)」*

Designer ツールの起動

このセクションでは、各 Designer ツールの起動方法について説明します。Designer ツールを使用すると、新規コンポーネントの作成や、既存のコンポーネントのオープンおよび変更を行うことができます。

注: Designer ツールのどれかを起動しようとしたときに「クラスが見つかりません」という内容のエラーが発生した場合は、System Manager を起動してから、デザイナー・ツールの起動を再試行する必要があります。ただし、ツールが起動した後は、System Manager を稼動したままにする必要はありません。

Business Object Designer

Business Object Designer を起動するには、以下のいずれかを実行します。

- WebSphere Business Integration システム・ビューで**「ビジネス・オブジェクト」**フォルダーを右マウス・ボタンでクリックし、コンテキスト・メニューから**「新規ビジネス・オブジェクトの作成」**を選択する。

- WebSphere Business Integration System Manager ビューで任意のフォルダーを選択し、以下のいずれかを実行する。
  - メニュー・バーから**「ツール」>「Business Object Designer」**を選択する。
  - **「Business Object Designer」**ツールバー・ボタンをクリックする。
  - キーボード・ショートカット **Ctrl+4** を使用する。
- **「スタート」>「プログラム」>「IBM WebSphere Business Integration Adapters」>「ツール」>「Business Object Designer」**を選択する。

Business Object Designer の詳細については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

Connector Configurator

Connector Configurator を起動するには、以下のいずれかの操作を実行します。

- WebSphere Business Integration システム・ビューで**「コネクター」**フォルダーを右マウス・ボタンでクリックし、コンテキスト・メニューから**「新規コネクターの作成」**を選択する。
- WebSphere Business Integration System Manager ビューで任意のフォルダーを選択し、以下のいずれかを実行する。
  - メニュー・バーから**「ツール」>「Connector Configurator」**を選択する。
  - **「Connector Configurator」**ツールバー・ボタンをクリックする。
  - キーボード・ショートカット **Ctrl+1** を使用する。
- **「スタート」>「プログラム」>「IBM WebSphere Business Integration Adapters」>「ツール」>「Connector Configurator」**を選択する。

Connector Configurator の詳細については、WebSphere Business Integration Adapter のユーザーズ・ガイドまたは「*コネクター開発ガイド (Java 用)*」または「*コネクター開発ガイド (C++ 用)*」 を参照してください。

新規コンポーネントの作成

183 ページの『Designer ツールの起動』で説明している各 Designer ツールを起動すると、新規コンポーネントを作成できます。例えば、Business Object Designer を起動すると新規ビジネス・オブジェクトが作成され、Connector Configurator を起動すると新規コネクターが作成されます。詳細については、これらの各ツールのマニュアルを参照してください。

既存のコンポーネントの変更

- ライブラリーのコンポーネントまたはユーザー・プロジェクトのコンポーネントへのショートカットをダブルクリックします。
- ライブラリーのコンポーネントまたはユーザー・プロジェクトのコンポーネントへのショートカットを選択し、以下のいずれかの操作を実行します。
  - 対応する Designer ツールを起動する。詳細については、183 ページの『Designer ツールの起動』を参照してください。
  - **Enter** を押す。
  - **Ctrl+E** を押す。
  - メニュー・バーから**「コンポーネント」>「定義の編集」**を選択する。

- ライブラリーのコンポーネントまたはユーザー・プロジェクトのコンポーネントへのショートカットを右マウス・ボタンでクリックし、コンテキスト・メニューから**「定義の編集」**を選択します。
- コンポーネントの Designer ツールを起動し (183 ページの『Designer ツールの起動』を参照)、ツールが起動したらコンポーネントを開きます。

パッケージからライブラリーへのコンポーネントのインポート

190 ページの『System Manager を使用したパッケージへのコンポーネントのエクスポート』の説明に従い、`.jar` ファイル・パッケージに統合コンポーネントをエクスポートできます。これにより、コンポーネントの環境間での移行、他の開発者との共用、およびテクニカル・サポートへのサブミットが容易になります。

パッケージから統合コンポーネント・ライブラリーにコンポーネントをインポートするには、以下の手順を実行します。

重要: インポートするパッケージのコンポーネントと同じ名前のコンポーネントが存在する場合、System Manager は警告を出さずに既存のコンポーネントを上書きします。

1. 統合コンポーネント・ライブラリーを右マウス・ボタンでクリックし、コンテキスト・メニューから**「リポジトリー・ファイルからインポート (Import from Repository File)」**を選択します。

   System Manager により、「リポジトリー・ファイルのインポート」ウィザードが表示されます (186 ページの図 31 を参照)。

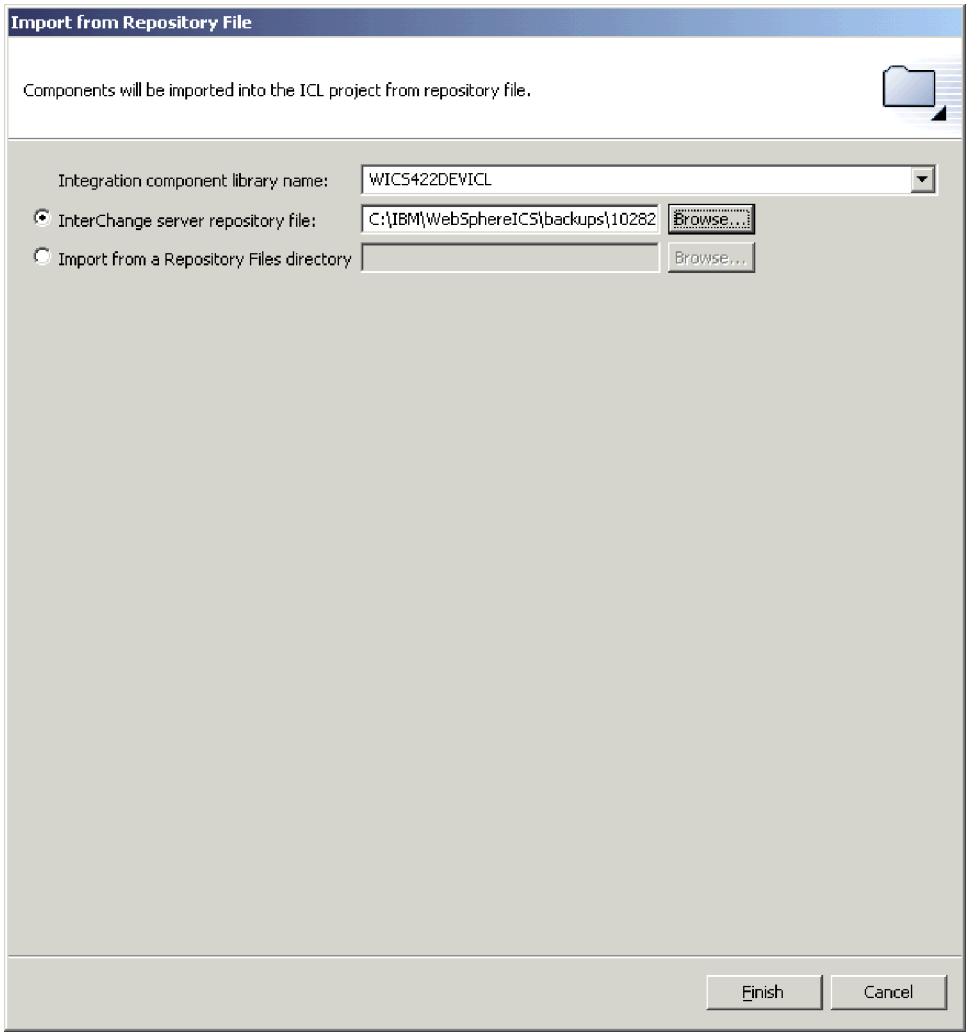


図 31. パッケージのインポート

2. 「リポジトリー・ファイルのインポート」画面で、「統合コンポーネント・ライブラリー名」ドロップダウン・メニューに、コンポーネントのインポート先のライブラリー名が含まれていることを確認します。

   コンポーネントのインポート先のライブラリー以外のライブラリーから「リポジトリー・ファイルからインポート (Import from Repository File)」ウィザードを起動した場合は、ウィザードを閉じて再起動しなくても、この方法でインポート先を変更できます。
3. 以下のいずれかを実行して、インポートするコンポーネントを指定します。
   - 単一のパッケージ・ファイルをインポートするには、インポートする `.jar` ファイルの絶対パスおよび名前を**「統合リポジトリー・ファイル (Integration repository file)」**フィールドに入力するか、**「ブラウズ」**をクリックしてファイルを選択します。
   - パッケージ・ファイルのディレクトリー全体をインポートするには、**「リポジトリー・ファイルからインポートするディレクトリー (Import from a Repository Files directory)」**フィールドにディレクトリーの絶対パスを入力するか、**「ブラウズ」**をクリックしてファイルを選択します。

4. **「終了」**をクリックします。

注: パッケージ・ファイルをインポートするには、ワークベンチの**「ファイル」> 「インポート」**メニュー項目は使用しないでください。「Zip ファイル」ウィザードは .jar 拡張子を持つアーカイブを処理でき、WebSphere Business Integration Adapter パッケージ・ファイルには .jar 拡張子が付いていますが、「Zip ファイル」ウィザードはパッケージ・ファイルを適切に処理しません。

ソリューションの処理

ユーザー・プロジェクトはソリューションとしてエクスポートできます。このアクションでは、ショートカットをユーザー・プロジェクトからコピーする他、統合コンポーネント・ライブラリーでショートカットが参照するコンポーネント定義もコピーします。これにより、ある環境から別の環境に全インターフェースまたはビジネス・インテグレーション・システムを容易に移行できます。

ソリューションのエクスポート

参照するユーザー・プロジェクトおよび統合コンポーネントをソリューションとしてエクスポートするには、以下の手順を実行します。

1. WebSphere Business Integration System Manager ビューで、**「ユーザー・プロジェクト」**フォルダーを展開し、**WAS** フォルダーを右マウス・ボタンでクリックし、コンテキスト・メニューから**「ソリューションのエクスポート」**を選択します。

   System Manager により、「ソリューションのエクスポート」ウィザードが表示されます (188 ページの図 32 を参照)。

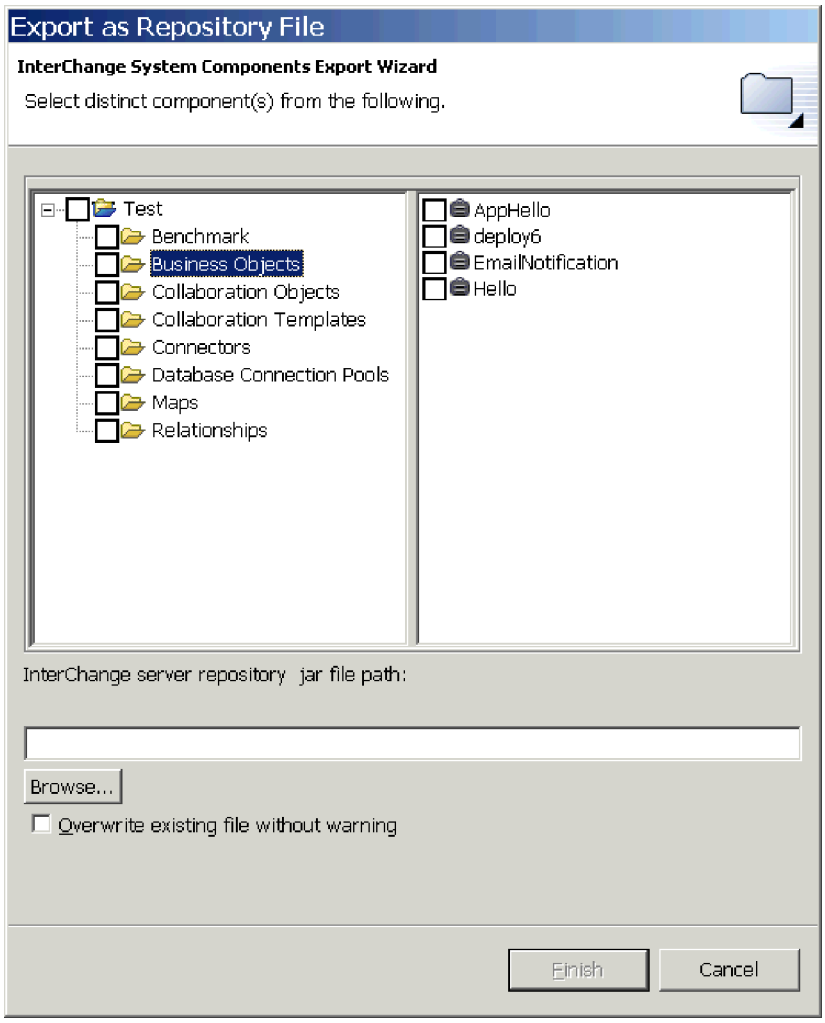


図 32. ソリューションのエクスポート

2. 以下の方法で、エクスポートするコンポーネントを選択します。
   - ユーザー・プロジェクトの横のチェック・ボックスを有効にして、プロジェクト内のすべてのコンポーネントを選択します。
   - コンポーネント・グループの横のチェック・ボックスを有効にして、グループ内のすべてのコンポーネントを選択します。
   - コンポーネント・グループを強調表示し、右側のペインで個々のコンポーネントの横にあるチェック・ボックスを有効にして、それらのコンポーネントを選択します。
3. ウィザード画面の下部にあるテキスト・フィールドにソリューションのエクスポート先ディレクトリーの絶対パスおよび名前を入力するか、**「ブラウズ」**をクリックして所定のディレクトリーにナビゲートします。
4. **「終了」**をクリックします。

   System Manager が以下の処理を実行し、ステップ 3 で指定されたディレクトリーにソリューションをエクスポートします。
   - ソリューションのエクスポート時に選択したユーザー・プロジェクトに、ショートカットを含む `User` ディレクトリーを作成します。

- ソリューションのエクスポート時に選択したユーザー・プロジェクトに、ショートカットによって参照される統合コンポーネント・ライブラリーのディレクトリーを含む System ディレクトリーを作成します。

5. エクスポート処理が正常に完了したことを示すプロンプトが出されたら、**「OK」**をクリックします。

ソリューションのインポート

ソリューションをインポートするには、以下の手順を実行します。

1. WebSphere Business Integration System Manager ビューで、**「ユーザー・プロジェクト」**フォルダーを展開し、**WAS** フォルダーを右マウス・ボタンでクリックし、コンテキスト・メニューから**「ソリューションのインポート」**を選択します。

   System Manager により、「ソリューションのインポート」ウィザードが表示されます (図 33 を参照)。

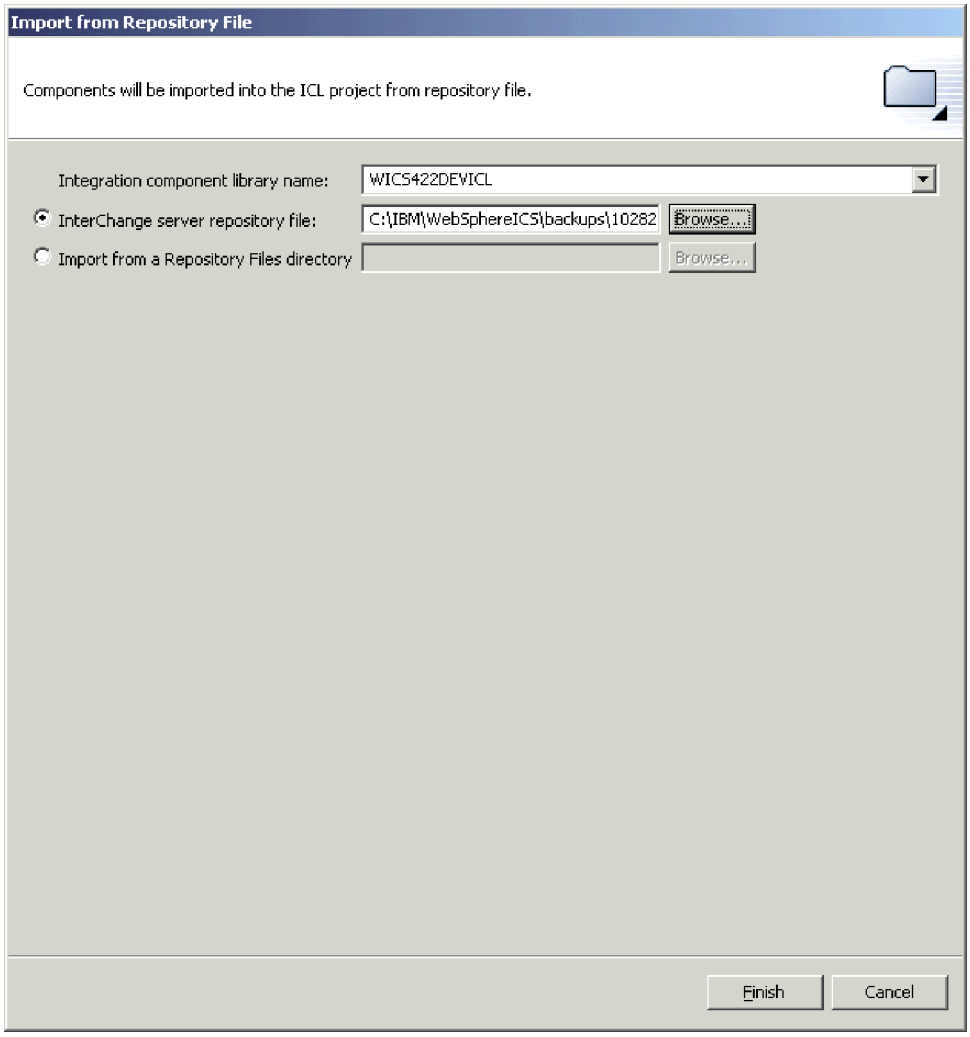


図 33. ソリューションのインポート

2. エクスポートしたソリューションが存在するディレクトリーの絶対パスおよび名前を**「ソリューション・ディレクトリー名 (Solution directory name)」**フィールドに入力するか、**「ブラウズ」**をクリックして所定のディレクトリーにナビゲートします。
3. **「終了」**をクリックします。

   System Manager が、エクスポートされたソリューションで定義されている統合コンポーネント・ライブラリーおよびユーザー・プロジェクトを環境に作成します。

System Manager を使用したパッケージへのコンポーネントのエクスポート

統合コンポーネントはパッケージ・ファイルにエクスポートできます。統合コンポーネントはリソースであり、最終的には、ファイル・システムに格納されるファイルとなります。詳細については、171 ページの『リソース』を参照してください。 System Manager は、コンポーネントをパッケージにエクスポートするときに以下のリソースを `.jar` (Java アーカイブ) ファイルに圧縮します。

- 定義ファイル (コンポーネント・タイプに応じた拡張子を付け、XML 形式で格納します)
- マップおよびコラボレーション・テンプレートの Java ソース・ファイル
- メッセージ・ファイル

コンポーネントをパッケージにエクスポートするには、以下の手順を実行します。

1. 統合コンポーネント・ライブラリー、またはエクスポートするコンポーネントが含まれるユーザー・プロジェクトを右マウス・ボタンでクリックし、コンテキスト・メニューから**「リポジトリー・ファイルとしてエクスポート (Export as Repository File)」**を選択します。

   System Manager により、「リポジトリー・ファイルのエクスポート」ウィザードが表示されます (191 ページの図 34 を参照)。

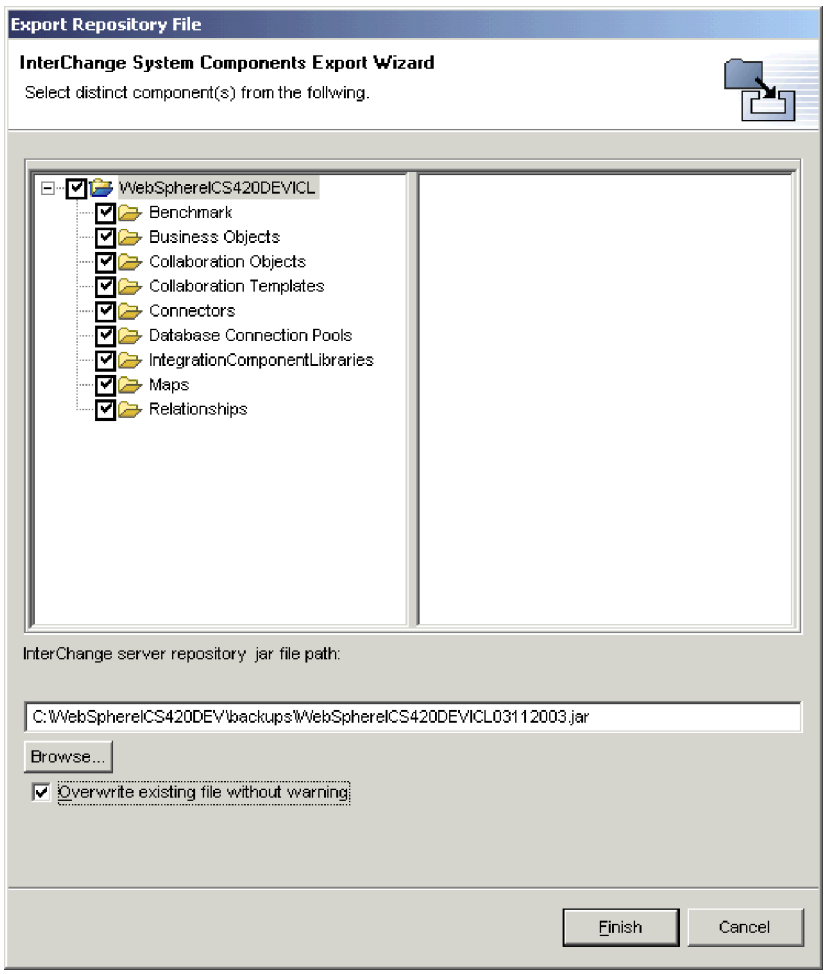


図 34. パッケージのエクスポート

2. 以下の方法で、エクスポートするコンポーネントを選択します。
   - 統合コンポーネント・ライブラリーまたはユーザー・プロジェクトの横のチェック・ボックスを有効にして、ライブラリーまたはプロジェクト内のすべてのコンポーネントを選択します。
   - コンポーネント・グループの横のチェック・ボックスを有効にして、グループ内のすべてのコンポーネントを選択します。
   - コンポーネント・グループを強調表示し、右側のペインで個々のコンポーネントの横にあるチェック・ボックスを有効にして、それらのコンポーネントを選択します。
3. コンポーネントのエクスポート先として既存の `.jar` ファイルを指定し、プロンプトを受け取ることなくファイルを上書きする場合は、**「警告せずに既存のファイルを上書き」**チェック・ボックスを有効にします。

   注: **「警告せずに既存のファイルを上書き」**チェック・ボックスを機能させるには、使用するファイルを指定する前に、このチェック・ボックスを有効にする必要があります (ステップ 4 (192 ページ) を参照)。System Manager は、既存のファイルが指定されていることを検出すると直ちにファイルを上書きするようプロンプトを出し、ウィザードが終了するまで待機しません。このため、このオプションはあらかじめ有効にしておく必要があります。

4. コンポーネントのエクスポート先の .jar ファイルの名前およびパスを**「リポジトリー jar ファイル・パス (Repository jar file path)」**フィールドに入力するか、**「ブラウズ」**をクリックして上書きするファイルを選択するか、またはディレクトリーをナビゲートしてファイル名を指定します。

   既存ファイルの名前およびパスを指定するときに**「警告せずに既存ファイルを上書き」**チェック・ボックスを有効にしなかった場合、既存のファイルを上書きするには、プロンプトが表示されたときに**「はい」**をクリックします。

   注: ファイルの名前およびパスをフィールドに入力する場合は、.jar 拡張子を含めなければ**「終了」**ボタンは使用可能になりません。
5. **「終了」**をクリックして、ウィザードを終了します。

依存関係および参照

統合コンポーネントは相互に依存し合ってビジネス・インテグレーション・システムでの役割を果たします。例えば、ビジネス・オブジェクト定義には他のビジネス・オブジェクト定義を子として含めることができ、コネクターのマップはサポートされるビジネス・オブジェクトに関連付けられます。システムを正常に機能させるには、これらの依存関係が満たされている必要があります。

コンポーネント間の関係について説明するときには、状況に応じて**依存関係**および**参照**という用語を使用します。例えば、コネクター定義が統合ブローカーとデータを交換するには、サポートされるビジネス・オブジェクト定義が存在している必要があります。この場合には、ビジネス・オブジェクト定義はコネクター定義の**依存関係**です。ビジネス・オブジェクト定義とコネクター定義の間に同じ関係がある場合でも、ビジネス・オブジェクトの文脈では、コネクターはビジネス・オブジェクト定義の**参照**の 1 つであり、そのビジネス・オブジェクトを子として含む他のビジネス・オブジェクト、そのビジネス・オブジェクトを変換するマップ、ポート定義用にビジネス・オブジェクトをサポートするコラボレーション・テンプレートなどを持ちます。

表 24 に、依存関係および参照となるコンポーネントをコンポーネント・タイプごとに示します。

表 24. 統合コンポーネントの依存関係および参照

コンポーネント	依存関係	参照
ビジネス・オブジェクト	• ビジネス・オブジェクト	• ビジネス・オブジェクト • マップ • コネクター • コラボレーション・テンプレート • コラボレーション・オブジェクト
コネクター	• ビジネス・オブジェクト • マップ	• コラボレーション・オブジェクト

表 24. 統合コンポーネントの依存関係および参照 (続き)

コンポーネント	依存関係	参照
マップ	• ビジネス・オブジェクト • マップ	• コネクター • コラボレーション・オブジェクト
コラボレーション・テンプレート	• ビジネス・オブジェクト	なし
コラボレーション・オブジェクト	• ビジネス・オブジェクト • マップ • コネクター • コラボレーション・テンプレート • コラボレーション・オブジェクト	• コラボレーション・オブジェクト

依存関係および参照の表示

System Manager を使用すると、統合コンポーネントの依存関係および参照を表示することができます。

コンポーネントの依存関係を表示するには、System Manager でコンポーネントを右マウス・ボタンでクリックし、コンテキスト・メニューから**「依存関係を表示」**を選択します。「依存関係ツリー」ウィザードが表示されます。詳細については、181 ページの『依存関係ツリーの使用』を参照してください。

コンポーネントの参照を表示するには、System Manager でコンポーネントを右マウス・ボタンでクリックし、コンテキスト・メニューから**「参照を表示」**を選択します。「オブジェクト参照」ウィンドウが表示されます。

複数のワークベンチ・リソースで使用可能な標準の操作

ワークベンチで行う多くの作業は処理している特定のリソースまたは作業している状況に依存しますが、すべてのリソースに同じように影響を及ぼす多くの操作も用意されています。このセクションでは、ワークベンチで実行可能な作業のうち、すべてのリソースに対して同様に機能するものについて説明します。

ソース・コード制御からワークスペースへのプロジェクトの追加

Rational ClearCase などのソース・コード制御システムから統合コンポーネント・ライブラリーおよびユーザー・プロジェクトをワークスペースに追加できます。

追加方法については、ソース・コード制御システム・プラグインの資料を参照してください。

ClearCase を使用して行う例については、「*WebSphere InterChange Server インプリメンテーション・ガイド*」を参照してください。

リソースの切り取り、コピー、および貼り付け

リソースの切り取り、コピー、および貼り付けは、System Manager とファイル・システムの両方で可能です。

System Manager でユーザー・プロジェクト、統合コンポーネント・ライブラリー、統合コンポーネント、ショートカット、またはフォルダーの切り取り、コピー、または貼り付けを行うには、リソースを右マウス・ボタンでクリックし、所定のメニュー項目を選択します。統合コンポーネントをコピーする場合は、同じライブラリーに貼り付ける (例えば、ビジネス・オブジェクト定義をコピーして同じライブラリーに貼り付け、別の名前を指定してテンプレートとして使用する) ことはできません。ただし、Designer ツールでコンポーネント定義を開き、「別名保管」操作を実行して新しい名前で同じライブラリーに保管することはできます。

ファイル・システムで統合コンポーネントまたはショートカットの切り取り、コピー、および貼り付けを行うには、Windows のエクスプローラーを起動し、プロジェクト・ディレクトリーの該当するサブディレクトリーにナビゲートし、コンポーネントの名前を共用するファイルをコピーし、宛先プロジェクト・ディレクトリー内の適切なサブディレクトリーに貼り付けます。

ファイル・システムで切り取り、コピー、および貼り付け操作を実行する場合は、必ず System Manager で統合コンポーネント・ライブラリーまたはユーザー・プロジェクトを最新表示させ、新しく追加したリソースが表示されるようにしてください。詳細については、『リソースの最新表示』を参照してください。

存在するユーザー・プロジェクトおよびライブラリーを指定するメタデータ参照がワークベンチで管理されているため、ユーザー・プロジェクト全体または統合コンポーネント・ライブラリー全体の切り取り、コピー、および貼り付けを行うことはできません。ワークスペース・ディレクトリーにフォルダーをコピーしても、メタデータ参照は更新されません。ただし、メタデータ・エントリーを満たすように System Manager で新規ユーザー・プロジェクトまたはライブラリーを作成してから、新規ライブラリーまたはユーザー・プロジェクトに対応するディレクトリーにコンポーネント定義のフォルダーを貼り付けることはできます。

リソースの最新表示

ファイル・システムでのファイルの切り取り、コピー、および貼り付けによってコンポーネント定義をライブラリーに追加したりショートカットをユーザー・プロジェクトに追加したりする場合は、必ず System Manager でライブラリーを最新表示させ、変更内容を反映させてください。

統合コンポーネント・ライブラリーまたはユーザー・プロジェクトを最新表示するには、System Manager でライブラリーまたはユーザー・プロジェクトを右マウス・ボタンでクリックし、コンテキスト・メニューから**「最新表示」**を選択します。

リソースの削除

ワークベンチ・リソースを削除するには、以下の手順を実行します。

1. System Manager でリソースを右マウス・ボタンでクリックしてコンテキスト・メニューから**「削除」**を選択するか、または System Manager でリソースを選択して**「削除」**キーを押します。

2. 「コンポーネントの削除」ダイアログが表示された場合は、**「OK」**をクリックします。

注: コンポーネントに依存関係が存在する場合は、そのコンポーネントを削除することはできません。

Eclipse ベースのワークベンチの使用

このセクションでは、ツールを効率的に使用するためにツール・フレームワークで実行する一部の操作について説明します。

パースペクティブのオープンおよびクローズ

このセクションでは、パースペクティブのオープン方法、およびクローズ方法について説明します。

パースペクティブのオープン

ワークベンチでパースペクティブを開くには、以下の手順を実行します。

1. ワークベンチのメニュー・バーから、**「ウィンドウ」>「パースペクティブを開く」>「その他」**を選択します。
2. 「パースペクティブの選択」ダイアログで開くパースペクティブを選択し、**「OK」**をクリックします。

System Manager パースペクティブを使用して統合コンポーネントを処理する他に、以下のセクションで説明しているパースペクティブも使用できます。

「リソース」パースペクティブ: このパースペクティブにより、統合コンポーネント・ライブラリーなどのプロジェクト・データを表すメタデータ・ファイルを直接処理できます。 196 ページの図 35に、「リソース」パースペクティブが開かれた状態を示します。このパースペクティブでは、ビジネス・オブジェクトを表す .xsd ファイルを「ナビゲーター」ビューから開き、その内容を「エディター」ビューに示しています。

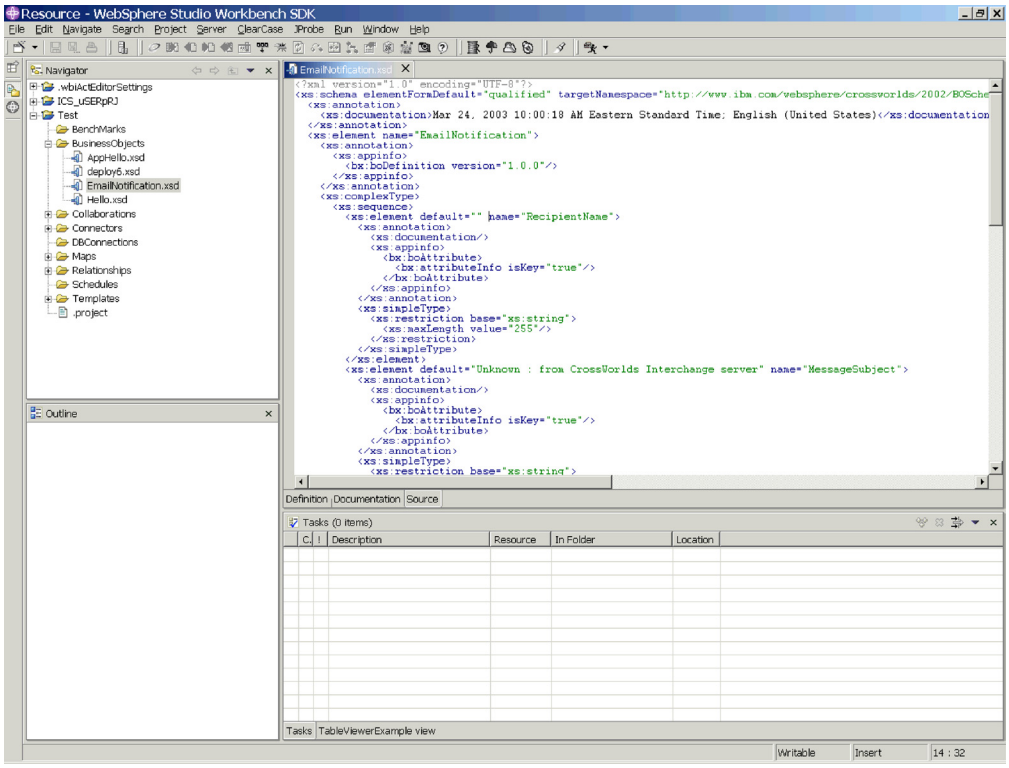


図 35. 「リソース」パースペクティブ

重要: 「リソース」パースペクティブから公開されるメタデータ・ファイルは、WebSphere Business Integration コンポーネントを定義します。 System Manager パースペクティブには、これらのコンポーネントをインターフェース経由で安全に処理するための方法が用意されています。直接メタデータ・ファイルを操作すると、コンポーネント定義を破壊してしまう危険があります。コンポーネント・メタデータ・ファイルを操作するのは、構造を熟知している場合、または定義のトラブルシューティングを行うためにテクニカル・サポートに連絡し、指示があった場合に限定してください。

Java パースペクティブ: このパースペクティブにより、Java ファイルのオーサリングに役立つエディターおよびビューを使用できます。Java プログラミングの大部分は Designer ツールで実行しますが、データ・ハンドラーなどの外部コンポーネントを記述したり、ユーティリティー・クラスを記述しなければならない場合もあります。このような場合に Java パースペクティブは非常に便利です。以下の図に、Java パースペクティブを示します。

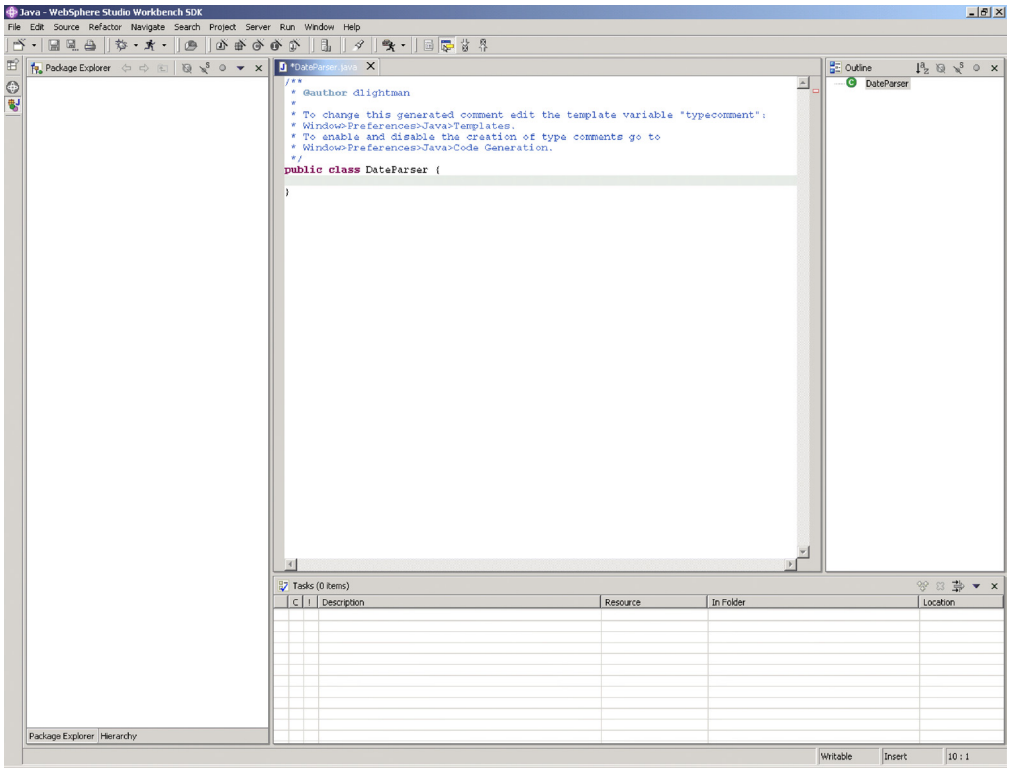


図 36. Java パースペクティブ

パースペクティブのクローズ

パースペクティブを閉じるには、以下の手順を実行します。

- 現在アクティブなパースペクティブを閉じるには、**「ウィンドウ」>「パースペクティブを閉じる」**を選択します。
- 現在開かれているパースペクティブをすべて閉じるには、**「ウィンドウ」>「すべてのパースペクティブを閉じる」**を選択します。
- パースペクティブ・ショートカット・バーにあるパースペクティブのアイコンを右マウス・ボタンでクリックし、コンテキスト・メニューから**「閉じる」**を選択すると、そのパースペクティブが閉じます。
- パースペクティブ・ショートカット・バーにあるパースペクティブのアイコンを右マウス・ボタンでクリックし、コンテキスト・メニューから**「すべて閉じる」**を選択すると、開いているすべてのパースペクティブが閉じます。

ビューの表示およびクローズ

WebSphere WorkBench パースペクティブおよび WebSphere Studio Application Developer Integration Edition パースペクティブに表示されるペインを制御できます。

ビューの表示

ビューを表示するには、以下の手順を実行します。

1. **「ウィンドウ」>「ビューの表示」>「その他」**を選択します。

2. ビュー・グループのフォルダー (**「WebSphere Business Integration Adapter Monitor のカテゴリー (WebSphere Business Integration Adapter Monitor Category)」** など) を展開します。
3. **「障害の発生しているキュー・マネージャー」**など、特定のビューを選択します。
4. **「OK」**をクリックします。

ビューのクローズ

ビューを閉じるには、以下の手順を実行します。

- ビューのタイトル・バーを右マウス・ボタンでクリックし、コンテキスト・メニューから**「閉じる」**を選択します。
- ビューのタイトル・バーにある「閉じる」ボタンをクリックします。

パースペクティブのカスタマイズ

パースペクティブをカスタマイズし、必要なパースペクティブ、ビュー、ウィザード、およびプラグイン・インターフェースをパースペクティブに組み込むことにより、頻繁に使用するエレメントを開いたり不要なエレメントを閉じたりする回数を最小限に抑えることができます。現在アクティブなパースペクティブをカスタマイズするには、以下の手順を実行します。

1. メニュー・バーから、**「ウィンドウ」>「パースペクティブのカスタマイズ」**を選択します。
2. カスタマイズするノードをクリックして展開します。
3. ノード・エレメントのチェック・ボックスを使用可能または使用不可に設定します。表 25 に、カスタマイズ可能なパースペクティブ・ノードおよびそのノードを使用可能化した場合の効果を示します。

表 25. カスタマイズ可能なパースペクティブ・ノード

パースペクティブ・ノードのカスタマイズ	結果
「ファイル」>「新規」	**「ファイル」>「新規」**メニューから項目を追加または除去します。
「ウィンドウ」>「パースペクティブを開く」	**「ウィンドウ」>「パースペクティブを開く」**メニューからパースペクティブを追加または除去します。
「ウィンドウ」>「ビューの表示」	**「ウィンドウ」>「ビューの表示」**メニューからビューを追加または除去します。
「その他」	メニュー・バーおよびツールバーから追加または除去します。例えば、**「ClearCase」**メニューを表示させるには、**「ClearCase」**チェック・ボックスを使用可能に設定する必要があります。

パースペクティブの保管

パースペクティブ構成を保管し、カスタマイズ内容を保存できます。パースペクティブを保管するには、以下の手順を実行します。

1. ツール・フレームワークのメニュー・バーから、**「ウィンドウ」>「パースペクティブの別名保管」**を選択します。

2. **「名前」**フィールドに、パースペクティブの名前を入力します。
3. **「OK」**をクリックします。

デフォルトでのパースペクティブの設定

デフォルトでは、ツール・フレームワークは「リソース」パースペクティブを開きます。主にツール・フレームワークを使用して統合コンポーネントを処理する場合は、System Manager をデフォルトのパースペクティブにすることができます。これを行うには、以下の手順を実行します。

1. ツール・フレームワークのメニュー・バーから、**「ウィンドウ」>「設定」**を選択します。
2. **「ワークベンチ」**ノードを展開します。
3. **「ワークベンチ」**ノードの下にある**「パースペクティブ」**ノードを選択します。
4. **「使用可能なパースペクティブ (Available Perspectives)」**リストから「System Manager 」を選択します。
5. **「デフォルトにする (Make Default)」**をクリックします。
6. **「OK」**をクリックします。

System Manager の設定の構成

System Manager の設定を構成するには、以下の手順を実行します。

1. ワークベンチのメニュー・バーから、**「ウィンドウ」>「設定」**を選択します。
2. **「System Manager の設定 (System Manager Preferences)」**を選択し、以下の手順を実行して使用可能な設定オプションを構成します。
   - 統合コンポーネント・ライブラリーからコンポーネントを削除するときにプロンプトが表示されないようにするには、「オブジェクトの削除について確認しない」ペインでコンポーネント・タイプに対応するチェック・ボックスを使用可能に設定します。
   - コピー操作時にコンポーネントとともにコンポーネントの依存関係もコピーする場合は、**「ディープ・コピー」**チェック・ボックスを使用可能に設定します。

     「ディープ・コピー」を使用可能に設定すると、ライブラリー間でビジネス・オブジェクト定義をコピーする場合に、例えば、含んでいる子ビジネス・オブジェクトもすべてコピーされます。しかし、**「ディープ・コピー」**を使用可能に設定しないと、ライブラリー間でビジネス・オブジェクト定義をコピーする場合に、そのビジネス・オブジェクト自体しかコピーされません。

     依存関係については、 192 ページの『依存関係および参照』を参照してください。
   - ファイルの名前およびパスを**「ログ・ファイル」**フィールドに入力するか、または**「参照」**をクリックしてファイルを選択します。System Manager でエラーが発生すると、指定したファイルにエラー情報が書き込まれます。メガバイト単位でログ・ファイルの最大サイズを指定するには、**「最大サイズ」**フィールドに数値を入力します。
   - 設定エレメントをデフォルト値に設定するには、**「デフォルトの復元」**をクリックします。

図 37 に、System Manager の設定インターフェースを示します。

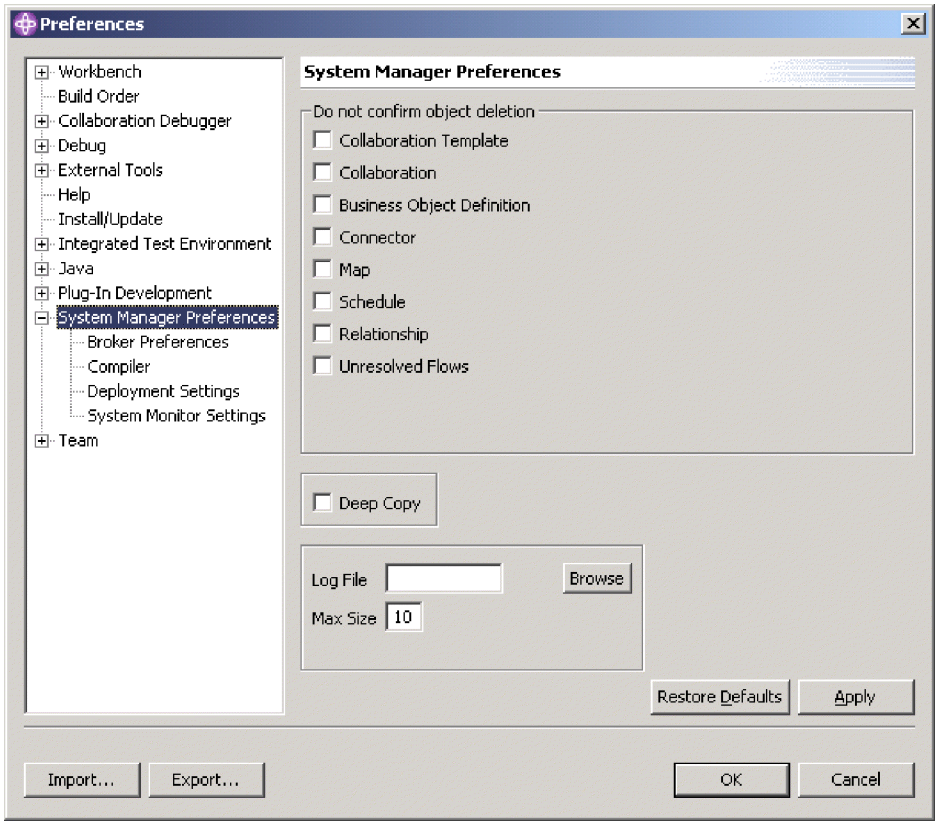


図 37. System Manager の設定

3. 「ブローカーの設定 (Broker Preferences)」インターフェースを使用すると、サポートされる WebSphere Message Broker と連携するように System Manager を構成できます。

   ブローカーの設定を行うには、以下の手順を実行します。

   a. **「System Manager の設定 (System Manager Preferences)」**を展開して、**「ブローカーの設定 (Broker Preferences)」**を選択します。

   b. WebSphere MQ Integrator Broker インポーター・ユーティリティーの絶対パスを**「Integrator Broker インポーター・パスを指定 (Specify the Integrator broker importer path)」**フィールドに入力するか、または**「参照」**をクリックしてディレクトリーを選択します。

   c. WebSphere Business Integration Message Broker インポーター・ユーティリティーの絶対パスを**「Message Broker インポーター・パスを指定 (Specify the Message broker importer path)」**フィールドに入力するか、または**「参照」**をクリックしてディレクトリーを選択します。

   d. WebSphere Business Integration Message Broker の Eclipse ワークスペース・ディレクトリーの絶対パスを**「Message Broker のワークスペース・ディレクトリーを指定 (Specify the Message broker workspace directory)」**フィールドに入力するか、または**「参照」**をクリックしてディレクトリーを選択します。

図 38 は、「ブローカーの設定 (Broker Preferences)」インターフェースを示しています。

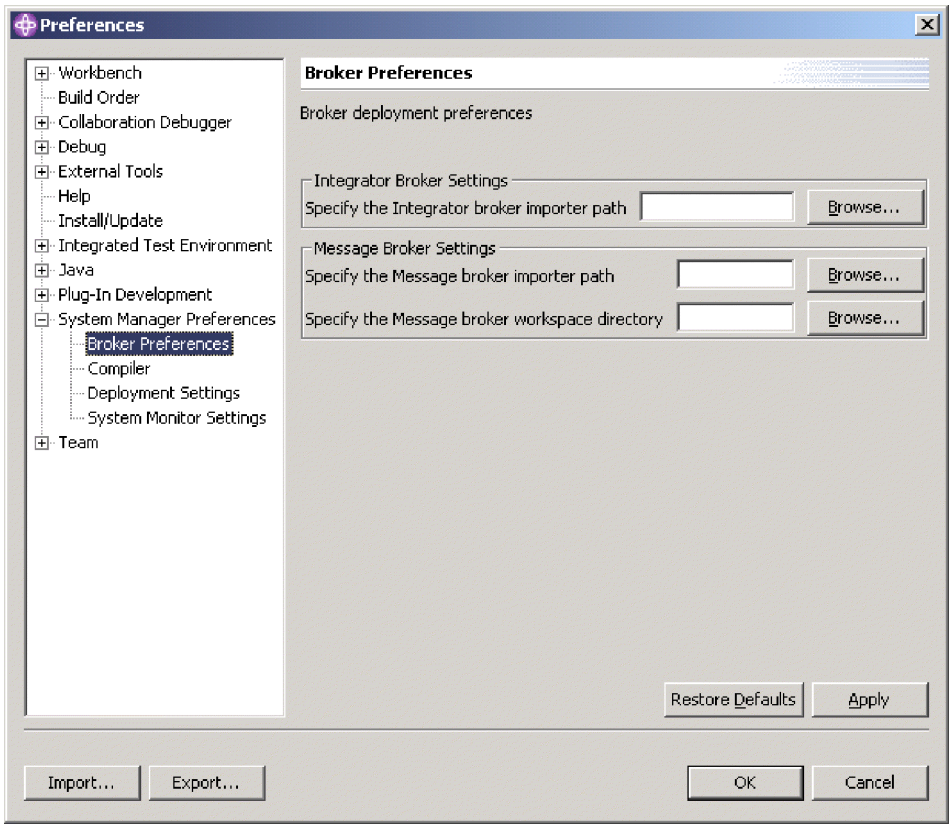


図 38. ブローカーの設定 (Broker Preferences)

注: 「コンパイラー (Compiler)」設定インターフェース、「配置設定 (Deployment Settings)」インターフェース、および「System Monitor の設定 (System Monitor Settings)」インターフェースは、統合ブローカーとして WebSphere Interchange Server を使用する場合に使用します。これらのインターフェースはその他の統合ブローカーには関係しません。

4. **「適用」**をクリックして設定を保管し、「設定」ダイアログでの作業を継続するか、または**「OK」**をクリックして設定を保管し、ダイアログを終了します。

System Manager で統合ブローカーに接続する際の問題のトラブルシューティング

接続に関する問題のトラブルシューティングを行うには、以下の可能性について調査します。

- インポーター・パスが指定されていることを確認してください。
- System Manager が正しいワークスペースに配置されていることを確認してください。

付録 F. Visual Test Connector の使用

Visual Test Connector は、コネクターの活動をシミュレートすることで、実際にコネクターを実行するという手間をかけずに、統合コンポーネントをテストできます。本章は、以下のセクションから構成されています。

- 『推奨されるテスト手順』
- 205 ページの『Test Connector の始動』
- 206 ページの『Test Connector のシャットダウン』
- 206 ページの『コネクター・プロファイルの作成と編集』
- 208 ページの『コネクターのエミュレート』
- 209 ページの『ビジネス・オブジェクトの処理』

推奨されるテスト手順

WebSphere Business Integration システムでコンポーネントをテストするための、推奨されるテスト手順は次のとおりです。

1. ご使用の統合ブローカーが InterChange Server の場合、System View ビューの使用を検討してください。送信したフローが成功したか失敗したかを判断するのに大変便利です。

   詳しくは、「*システム管理ガイド*」を参照してください。
2. Test Connector を、ソース・コネクターをエミュレートするようにセットアップする。
   a. 205 ページの『Test Connector の始動』の説明に従って、Test Connector を起動する。
   b. インターフェース内のソース・コネクター用プロファイルを、206 ページの『新規プロファイルの作成』の説明に従って作成する。
   c. 208 ページの『コネクターのエミュレート』の説明に従って、Test Connector をエージェントに接続し、ソース・コネクターのエミュレーションを開始する。
3. Test Connector のインスタンスを、インターフェースに接続された各宛先コネクターをエミュレートするようにセットアップする。
   a. 205 ページの『Test Connector の始動』の説明に従って、Test Connector を起動する。
   b. 宛先コネクター用のプロファイルを、 206 ページの『新規プロファイルの作成』に示す方法で作成する。
   c. 208 ページの『コネクターのエミュレート』の説明に従って、Test Connector をエージェントに接続し、宛先コネクターのエミュレーションを開始する。
   d. インターフェースに接続されたすべての宛先コネクターについて、上記の 3a から 3c を繰り返す。
4. 各「Test Connector」ウィンドウで、エミュレートしているコネクターを識別しやすいように、画面上の Test Connector のインスタンスを配置する。

例えば、204 ページの図 39 では、ソース・テスト・コネクターを宛先テスト・コネクターの左側に配置しています。

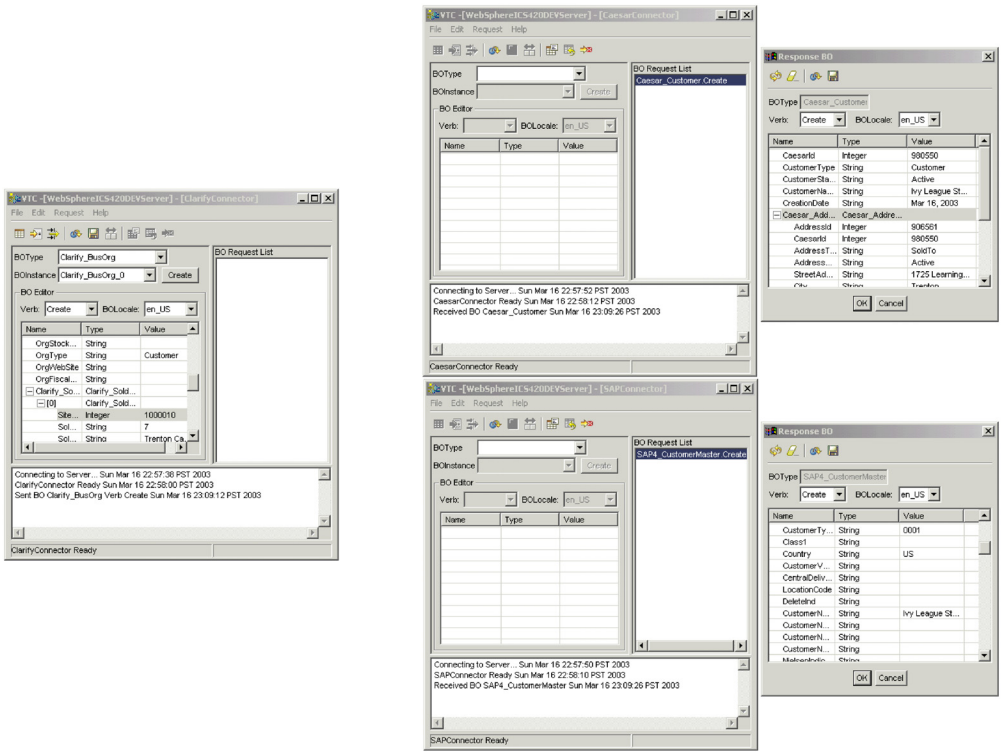


図 39. Test Connector のソース・インスタンスおよび宛先インスタンス

5. ソース・コネクターから要求ビジネス・オブジェクトを送信する。ソース・テスト・コネクターで、以下の手順を実行します。
   a. テストを必要とするインターフェースにより管理されるビジネス・オブジェクトを、 209 ページの『要求ビジネス・オブジェクトの作成』に示す方法により作成する。
   b. 214 ページの『ビジネス・オブジェクトの保管』の説明に従って、このビジネス・オブジェクトをファイルに保管し、以降のテストでも使用できるようにする。
   c. ビジネス・オブジェクトを、 210 ページの『要求ビジネス・オブジェクトの送信』に示す方法により送る。
6. 要求ビジネス・オブジェクトに対する宛先コネクターからの応答をシミュレートする。宛先「Test Connector」ウィンドウで、以下の手順を実行します。
   a. ビジネス・オブジェクトを、 215 ページの『要求ビジネス・オブジェクトの受け入れ』に示す方法により受け付ける。
   b. ビジネス・オブジェクトを、 216 ページの『応答ビジネス・オブジェクトの送信』に示す方法により応答として送る。
7. ステップ 5 から 6 を、各インターフェースのテストに必要な回数だけ繰り返す。

Test Connector の始動

Test Connector を始動するには、統合ブローカーの種類に応じて、次のいずれかの方法を実行します。

- 統合ブローカーが InterChange Server である場合は、**「スタート」>「プログラム」>「IBM WebSphere InterChange Server」>「IBM WebSphere Business Integration Toolset」>「開発」>「Test Connector」**を選択します。
- ご使用の統合ブローカーが WebSphere Application Server または WebSphere Message Broker (WebSphere MQ Integrator、WebSphere MQ IntegratorBroker、または WebSphere Business Integration Message Broker) の場合、**「スタート」>「プログラム」>「IBM WebSphere Business Integration Adapters」>「ツール」>「Test Connector」**を選択します。

図 40 に、始動後の Test Connector を示します。

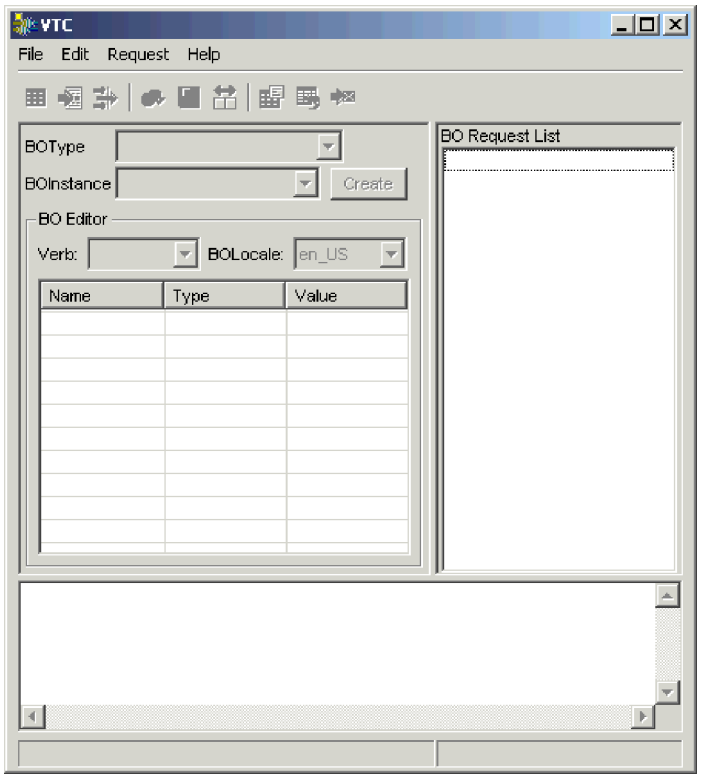


図 40. Test Connector

「Test Connector」ウィンドウには、以下のペインがあります。

- 「サポートされているビジネス・オブジェクト」ペインでは、送信するビジネス・オブジェクトのインスタンスを作成できます。
- 「ビジネス・オブジェクト要求リスト」ペインには、コネクターが受信したすべてのビジネス・オブジェクト要求が表示されます。
- 「出力」ペインには、ビジネス・オブジェクトが送信された時刻などの、Test Connector の操作に関するメッセージが表示されます。

Test Connector のシャットダウン

Test Connector をシャットダウンし、コネクター・エージェントのエミュレーションを停止させるには、メニュー・バーから「**ファイル**」>「**終了**」を選択します。「シャットダウン」プロンプトが表示された場合は、「**はい**」をクリックします。

コネクター・プロファイルの作成と編集

Test Connector は、コネクターをエミュレートするために必要となる情報を保管するために、プロファイルを使用します。エミュレートするコネクターごとにプロファイルを作成する必要があります。既存のプロファイルを編集したり、削除したりすることも可能です。

ファイルへのコネクター定義の保管

Test Connector を使用してコネクターをエミュレートするには、コネクター定義をファイルに保管しておく必要があります。コネクター定義をファイルに保管するには、以下の手順を実行します。

1. Connector Configurator で、コネクター定義を開く。
2. メニュー・バーから「**ファイル**」>「**別名保管**」>「**ファイルに**」を選択する。
3. ファイルの保管先とするディレクトリーに移動し、「**ファイル名**」フィールドに名前を入力する。「ファイルの種類 (Save as type)」ドロップダウン・メニューに値「**構成ファイル (*.cfg)**」が表示されていることを確認して、「**保管**」をクリックする。

   Connector Configurator は、指定した名前でコネクター定義をファイルに保管します。

新規プロファイルの作成

プロファイルは、Test Connector でエミュレートするすべてのコネクターについて作成する必要があります。プロファイルでは、コネクターの名前、使用する構成ファイル、コネクターが通信する統合ブローカーのタイプなどの情報を指定します。新規コネクター・プロファイルを作成する手順は次のとおりです。

1. メニュー・バーから「**ファイル**」>「**プロファイルを作成/選択**」を選択し、「コネクター・プロファイル」ウィンドウを表示する。
2. 「コネクター・プロファイル」ウィンドウで、メニュー・バーから「**ファイル**」>「**新規プロファイル**」を選択する。
3. 「新規プロファイル」ウィンドウで「**参照**」をクリックし、『ファイルへのコネクター定義の保管』で用意したコネクターの構成ファイルに移動する。
4. 「**コネクター名**」フィールドに、コネクターの名前を入力する。コネクター定義の名前は、統合ブローカーのリポジトリーに存在するものと完全に同じものを入力する必要があります。例えば、JText 用アダプターの場合は、`JTextConnector` と入力します。その際、`JText` という単語と `Connector` という単語の間にスペースを入れないようにし、すべての文字の大文字と小文字を正しく入力する必要があります。
5. 「**ブローカー・タイプ**」ドロップダウン・メニューで、適切な統合ブローカー (`ICS`、`WMQI`、または `WAS`) を選択する。

注: ご使用のブローカーが WebSphere Message Broker の場合、WMQI を選択してください。

6. ステップ 5 (206 ページ) で、ブローカーのタイプとして ICS を選択した場合は、以下の作業も行なう必要があります。
   a. 「**サーバー**」フィールドに、InterChange Server インスタンスの名前を入力する。

      名前は正確に入力してください。この名前では大文字小文字が区別されます。名前が不正確な場合には、Test Connector は InterChange Server と通信できません。

   b. 「**パスワード**」フィールドに、admin ユーザー・アカウントのパスワードを入力する。デフォルトのパスワードは null です。

      図 41 に、「新規プロファイル」ウィンドウを示します。

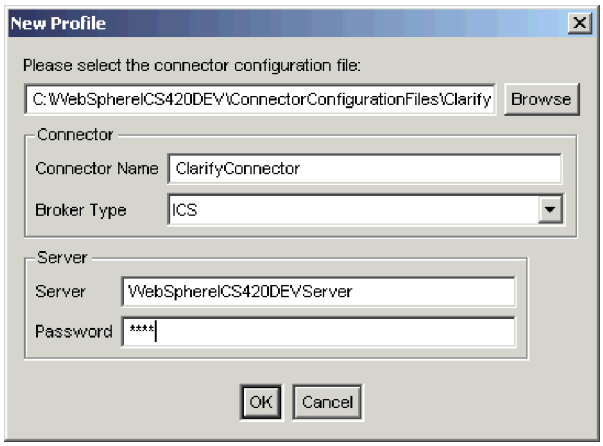


図 41. 新規コネクター・プロファイルの作成

7. 「**OK**」をクリックして、「新規プロファイル」ウィンドウを閉じる。

「コネクター・プロファイル」ウィンドウの「**コネクター**」列にはコネクターの名前が、「**サーバー**」列には InterChange Server インスタンスの名前 (統合ブローカーが ICS の場合) が、「**構成ファイル**」列にはコネクター構成ファイルのパスと名前が表示されます。

208 ページの図 42 に「コネクター・プロファイル」ウィンドウを示します。ここでは、ClarifyConnector のプロファイルが InterChange Server インスタンスと通信するように構成され、JTextConnector のプロファイルが WMQIB サーバーと通信するように構成されています。

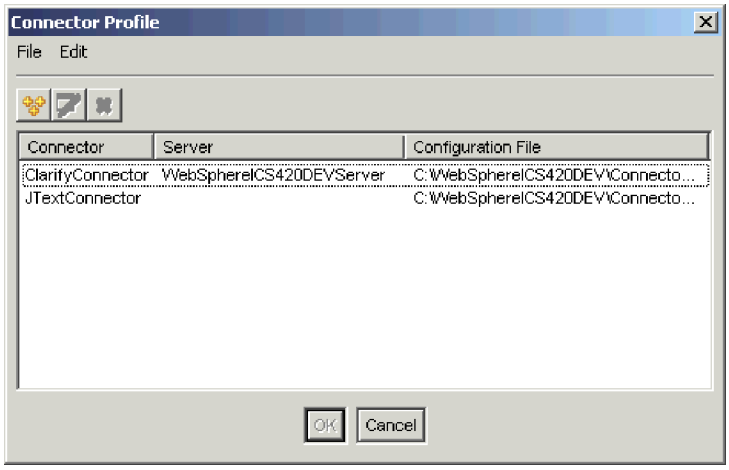


図 42. 「コネクター・プロファイル」ウィンドウ

8. 「**OK**」をクリックして、「コネクター・プロファイル」ウィンドウを閉じる。

プロファイルの編集

既存のコネクター・プロファイルに変更を加える手順は次のとおりです。

1. Test Connector のメニュー・バーから「**ファイル**」>「**プロファイルを作成/選択**」を選択するか、またはキーボード・ショートカット **Ctrl+N** を使用して、「コネクター・プロファイル」ウィンドウを表示する。
2. 「コネクター・プロファイル」ウィンドウで、編集するプロファイルを選択し、次にメニュー・バーから「**編集**」>「**プロファイルを編集**」を選択する。
3. 「新規プロファイル」ウィンドウの各フィールドに新しい値を入力し、「**参照**」ボタンを使用して、編集作業の必要に応じて構成ファイルを切り替える。
4. 「**OK**」をクリックして、「新規プロファイル」ウィンドウを閉じる。

プロファイルの削除

コネクター・プロファイルを削除するには、以下の手順を実行します。

1. Test Connector のメニュー・バーから「**ファイル**」>「**プロファイルを作成/選択**」を選択するか、またはキーボード・ショートカット **Ctrl+N** を使用して、「コネクター・プロファイル」ウィンドウを表示する。
2. 「コネクター・プロファイル」ウィンドウで、削除するプロファイルを選択し、次にメニュー・バーから「**編集**」>「**プロファイルを削除**」を選択する。

コネクターのエミュレート

コネクター用のプロファイルを作成した後、そのプロファイルを使用して、Test Connector をエージェントに接続できます。Test Connector をエージェントに接続すると、Test Connector は、選択したプロファイルで定義されているコネクターのエミュレーションを開始します。

Test Connector をエージェントに接続するには、以下の手順を実行します。

1. Test Connector のメニュー・バーから、「**ファイル**」>「**プロファイルを作成/選択**」を選択する。

2. 「コネクター・プロファイル」ウィンドウで、これから開くコネクター用プロファイルのコネクター名を選択する。
3. **「OK」**をクリックする。
4. メニュー・バーから、「**ファイル**」>「**接続**」を選択する。

   Test Connector は、コネクターのエミュレーションを試行する際に、「出力」ペインにメッセージを表示します。接続が完了すると、準備が完了したことを示すメッセージを「出力」ペインに表示し、「サポートされているビジネス・オブジェクト」ペインに「**BOType**」リストを取り込みます。

ビジネス・オブジェクトの処理

ビジネス・プロセス・インターフェースが正しく開発されたことを検証するには、ビジネス・オブジェクトが正常に交換および処理されることを確認する必要があります。このセクションでは、以下の方法について説明します。

- ビジネス・オブジェクト・テスト・データの作成、変更、削除、および保管
- ビジネス・オブジェクトの属性値を比較しての、処理中に行われた変更の迅速かつ容易な確認
- ビジネス・オブジェクトの送受信

要求ビジネス・オブジェクトの処理

要求ビジネス・オブジェクトは、インターフェースを起動するイベントのソース・コネクターを Visual Test Connector がエミュレートしている際に、Test Connector から送信されるビジネス・オブジェクトです。要求ビジネス・オブジェクトの処理では、ビジネス・オブジェクト・インスタンスを作成し、そのインスタンスにデータを取り込み、要求を送信します。

要求ビジネス・オブジェクトの作成

Test Connector で新しいビジネス・オブジェクトを作成するには、以下の手順を実行します。

1. 「サポートされているビジネス・オブジェクト」ペインで、作成するビジネス・オブジェクトの名前を「**BOType**」ドロップダウン・メニューから選択する。
2. 「**BOInstance**」フィールドの隣にある「**作成**」をクリックする。
3. 「新規インスタンス (New Instance)」ダイアログが表示されたら、インスタンスの名前を「**名前を入力してください**」フィールドに入力する。
4. 「**動詞**」ドロップダウン・メニューから、必要な動詞を選択する。
5. 「**BOLocale**」ドロップダウン・メニューから、必要なロケールを選択する。
6. 213 ページの『ビジネス・オブジェクト属性の値の設定』に示す方法により、トップレベル・オブジェクト内の単純属性および子ビジネス・オブジェクトの値を指定する。

   210 ページの図 43 に、ビジネス・オブジェクト `Caesar_Customer` を示します。このビジネス・オブジェクトは `Create` 動詞を持ち、ロケールは `en_US` であり、各単純属性に値が指定されており、`Caesar_Address` 子ビジネス・オブジェクトの単一のインスタンスがあります。

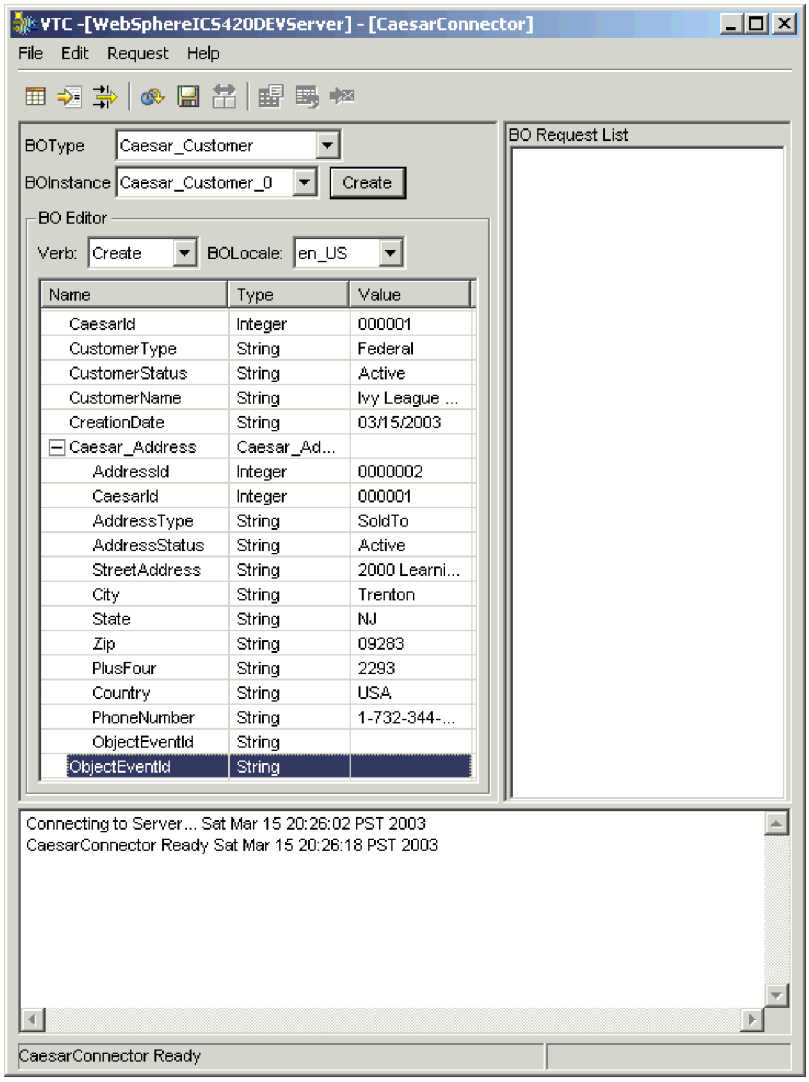


図 43. ビジネス・オブジェクトへのデータの取り込み

7. 「**OK**」をクリックする。

要求ビジネス・オブジェクトの送信

ビジネス・オブジェクトを作成またはロードし、その属性の値を指定した後、このビジネス・オブジェクトを要求としてインテグレーション・ブローカーに送信するには、いくつかの方法があります。

非同期モードでの要求ビジネス・オブジェクトの送信: ソース・コネクターは、要求ビジネス・オブジェクトを非同期モードで送ったとき、応答ビジネス・オブジェクトが戻ることを期待しません。要求ビジネス・オブジェクトがディスパッチされてしまえば、トランザクションにおけるソース・コネクターの役割は終了します。応答ビジネス・オブジェクトは、通常はインテグレーション・ブローカーが処理します。Test Connector のデフォルトのモードは非同期です。

ビジネス・オブジェクトを非同期モードで送信する手順は次のとおりです。

1. メニュー・バーから、「**要求**」>「**モード**」>「**非同期**」を選択する。

注: Test Connector は、デフォルトでは「非同期」モードで作動するため、このステップは、以前にコネクターから同期要求を送信していた場合にのみ行なう必要があります。また、各要求を送信する前にモードを設定する必要はありません。

2. メニュー・バーから、「**要求**」>「**送信**」を選択する。

コネクター定義で指定したブローカーが InterChange Server の場合は、ビジネス・オブジェクト要求がサーバーに送信され、処理されます。

コネクター定義で指定したブローカーが、サポートされるメッセージ・ブローカーか WebSphere Application Server の場合には、ビジネス・オブジェクトは `RequestQueue` 標準プロパティーで指定しているキューに置かれます。

同期モードでの要求ビジネス・オブジェクトの送信: ソース・コネクターは、要求ビジネス・オブジェクトを同期モードで送信すると、宛先アプリケーションが要求を処理した後、インテグレーション・ブローカーから応答ビジネス・オブジェクトが戻ることを期待します。同期モードでは、Test Connector は応答ビジネス・オブジェクトを、ソース・コネクターの Synchronous Request Queue プロパティーで指定されたキューに入れます。Test Connector のデフォルトのモードは非同期です。

1. メニュー・バーから「**要求**」>「**モード**」>「**同期**」を選択して、Test Connector を同期モードに設定する。
2. メニュー・バーから、「**要求**」>「**送信**」を選択する。
3. コネクター定義で指定されているブローカーが InterChange Server の場合は、「コラボレーションの選択」ダイアログが表示される。ビジネス・オブジェクトの送信先とするコラボレーションを、「**コラボレーション**」ドロップダウン・メニューから選択し、「**OK**」をクリックする。

コネクター定義で指定したブローカーが InterChange Server の場合は、処理を選択したコラボレーション・オブジェクトの構成済みポートにビジネス・オブジェクト要求が送信されます。

コネクター定義で指定したブローカーが サポートされるメッセージ・ブローカーか、WebSphere Application Server の場合には、ビジネス・オブジェクトは `SynchronousRequestQueue` 標準プロパティーで指定しているキューに置かれます。

バッチ・モードでの要求ビジネス・オブジェクトの送信: バッチ・モードの場合、Test Connector では、送信する特定のビジネス・オブジェクトのインスタンス数を指定できます。また、それぞれのインスタンスごとに固有値に設定する必要のあるトップレベル・オブジェクトの 1 つの属性 (例えば、基本キー属性) も指定できます。Test Connector は、指定された回数だけビジネス・オブジェクトをコピーし、指定された 1 つの属性の値を増分し、それぞれのビジネス・オブジェクトを送信します。このオプションにより、多数のビジネス・オブジェクトを迅速かつ容易に作成することができます。

選択された属性が、識別関係の一環として動的相互参照に関与しているキー・フィールドである場合は、初期値も後続する値も固有であることを保証する必要があります。この条件が満たされていないと、相互参照のロジックが成立せず、要求ビジネス・オブジェクトは失敗します。

値が固有であることを確認するためには、関係マネージャーを使用するか、次の手順により関係参加項目のテーブルに対して SQL ステートメントを実行します。

- 参加項目の現在の最高値を判別し、それよりも大きな値を初期値フィールドに設定する。バッチ送信の最初のビジネス・オブジェクト・インスタンスおよび後続のインスタンスはすべて固有となります。
- 参加項目の既存テーブル・エントリーを削除することにより、どのテーブル・エントリーもバッチ・ビジネス・オブジェクトのどれとも属性値が同じにならないようにする。

バッチ・モードでビジネス・オブジェクトを送信する手順は次のとおりです。

1. 送信するビジネス・オブジェクトの名前を、「**BOType**」ドロップダウン・メニューから選択する。
2. メニュー・バーから、「**要求**」>「**バッチを送信**」を選択する。
3. 「バッチ・モード」ウィンドウで、**「動詞」**ドロップダウン・メニューから、必要な動詞を選択する。
4. 「**BOLocale**」ドロップダウン・メニューから、必要なロケールを選択する。
5. **「属性」** リストから、バッチ内の各ビジネス・オブジェクト要求ごとに増分させる、トップレベル・ビジネス・オブジェクトの属性を選択する。

   通常は、基本キーなど、ビジネス・オブジェクトを一意的に特定する属性を選択します。
6. 「**初期値**」フィールドに、増加させる属性の初期値を入力する。
7. 「**ビジネス・オブジェクト数**」フィールドに、生成して送信するビジネス・オブジェクト・インスタンスの数を入力する。
8. **「OK」**をクリックします。

   Test Connector は、指定された数のビジネス・オブジェクトを生成します。これらのビジネス・オブジェクトは、インスタンスごとに値が増分されるように指定された 1 つの属性を除いて、すべて同一です。

   コネクター定義で指定したブローカーが InterChange Server の場合は、ビジネス・オブジェクト要求がサーバーに送信され、処理されます。

   コネクター定義で指定したブローカーが、サポートされるメッセージ・ブローカーか WebSphere Application Server の場合には、ビジネス・オブジェクトは `RequestQueue` 標準プロパティーで指定しているキューに置かれます。

213 ページの図 44 では、以下のようなバッチ・モード構成を示しています。

- 50 個のビジネス・オブジェクトを送信します。
- 属性 `OrgObjid` の値を増分します。
- 属性の開始値は 100001 です。

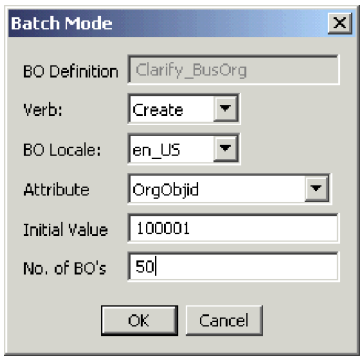


図 44. 「バッチ・モード」ウィンドウ

ビジネス・オブジェクト属性の値の設定

以下の各セクションでは、ビジネス・オブジェクト・インスタンスの単純属性および複合属性の値を設定するためのさまざまな方法について説明します。

- 『単純属性の値の設定』
- 『子ビジネス・オブジェクトの追加』
- 214 ページの『子ビジネス・オブジェクトの除去』
- 214 ページの『子ビジネス・オブジェクトの動詞の設定』

単純属性の値の設定

単純属性の値を設定するには、「**値**」列でその属性のセルをクリックして、値を入力します。

子ビジネス・オブジェクトの追加

子ビジネス・オブジェクトのインスタンスを追加するには、その子ビジネス・オブジェクトを表す属性上で右マウス・ボタンをクリックし、コンテキスト・メニューから「**インスタンスを追加**」を選択します。

子ビジネス・オブジェクトを表す属性の隣に正符号 (+) が追加され、少なくとも 1 つの子ビジネス・オブジェクト・インスタンスが存在することを示します。子オブジェクト属性を展開すると、各インスタンスに対応した番号付き項目が表示されます。個々のインスタンスにもその隣に正符号 (+) が付いているため、それらを展開して属性に値を設定できます。

子ビジネス・オブジェクトのインスタンスをさらに追加するには、子オブジェクトを表す属性を右マウス・ボタン・クリックし、コンテキスト・メニューから「**インスタンスを追加**」を選択します。

注: 子ビジネス・オブジェクトを参照する属性の「**カード**」プロパティーが値 1 (カーディナリティーが単一であることを示す) に設定されている場合、追加できる子オブジェクトのインスタンスは 1 つのみです。

子ビジネス・オブジェクトの除去

子ビジネス・オブジェクトのインスタンスを削除するには、そのインスタンスの上で右マウス・ボタンをクリックし、コンテキスト・メニューから「**インスタンスを削除**」を選択します。

子ビジネス・オブジェクトのすべてのインスタンスを削除するには、子ビジネス・オブジェクトを表す属性を右マウス・ボタン・クリックし、コンテキスト・メニューから「**全インスタンスを削除**」を選択します。

子ビジネス・オブジェクトの動詞の設定

子ビジネス・オブジェクトの動詞を設定することにより、設定された値がビジネス・オブジェクトに及ぼす影響をテストすることができます。これは、子オブジェクトの相互参照が絡んでいるロジックのトラブルシューティングを実行するときに役立ちます。

子ビジネス・オブジェクト・インスタンスの動詞を設定するには、そのインスタンスを右マウス・ボタン・クリックし、コンテキスト・メニューから「**動詞を設定**」を選択します。「動詞の選択」プロンプトが表示されたら、必要な動詞を選択し、「**OK**」をクリックします。

「応答ビジネス・オブジェクト」ツールバーの使用

宛先コネクターが受信したビジネス・オブジェクトの属性は、応答として送信する前に編集することができます。編集する際に使用する「応答ビジネス・オブジェクト」ダイアログのツールバーには、ビジネス・オブジェクトの値を設定するために使用できる、いくつかのツールバー・ボタンがあります。詳しくは、 216 ページの『応答ビジネス・オブジェクトの編集』を参照してください。

ビジネス・オブジェクトの保管

Test Connector では、ビジネス・オブジェクトを保管して、後のテストで使用したり、テクニカル・サポートに提供したり (問題のトラブルシューティングに役立てるため)、応答データとして使用したりすることができます。ユーザーが作成したビジネス・オブジェクトや、宛先コネクターの「Test Connector」ウィンドウに要求として表示されたビジネス・オブジェクトを含め、すべてのビジネス・オブジェクトを保管できます。デフォルトでは、ビジネス・オブジェクトはビジネス・オブジェクト拡張子 (`.bo`) の付いたファイルに保管されます。

必要に応じて、テスト・データ・ファイル専用のディレクトリーやディレクトリー構造 (各インターフェースまたは各コネクターに対応したサブディレクトリーを持つ) を作成することをお勧めします。この構造により、必要なファイルが見つけやすくなり、テストの効率が上がります。さらに、ビジネス・オブジェクトのテスト・データ・ファイルには、ビジネス・オブジェクト定義自体と同じ名前を付けることをお勧めします。

要求ビジネス・オブジェクトの保管

要求として作成したビジネス・オブジェクト・インスタンスを保管するには、以下の手順を実行します。

1. 保管したいビジネス・オブジェクトを選択する。

2. メニュー・バーから、「**編集**」>「**ビジネス・オブジェクトを保管**」を選択する。
3. 目的のディレクトリーにナビゲートし、「**ファイル名**」フィールドにファイルの名前を指定する。
4. 「**保管**」をクリックする。

応答ビジネス・オブジェクトの保管

Test Connector の宛先インスタンスが受信し、応答として送信されるビジネス・オブジェクト・インスタンスを保管するには、以下の手順を実行します。

1. 「ビジネス・オブジェクト要求リスト」ペインで、ビジネス・オブジェクト・インスタンスを選択する。
2. メニュー・バーから、「**要求**」>「**応答を編集**」を選択する。
3. 「**ビジネス・オブジェクトを保管**」をクリックする。
4. 目的のディレクトリーにナビゲートし、「**ファイル名**」フィールドにファイルの名前を指定する。
5. 「**保管**」をクリックする。

ビジネス・オブジェクトのロード

ファイルに保管されたビジネス・オブジェクトをロードする手順は次のとおりです。

1. Test Connector のメニュー・バーから、「**編集**」>「**ビジネス・オブジェクトをロード**」を選択する。
2. ビジネス・オブジェクトのテスト・データ・ファイルに移動し、それを開く。
3. 「新規インスタンス (New Instance)」ダイアログが表示されたら、インスタンスの名前を「**名前を入力してください**」フィールドに入力する。
4. **「OK」**をクリックする。

ビジネス・オブジェクトの削除

ビジネス・オブジェクトを Test Connector から削除するには、メニュー・バーから「**編集**」>「**ビジネス・オブジェクトを削除**」を選択します。

注: この操作では、Test Connector からビジネス・オブジェクトのみが削除されます。コネクターによるビジネス・オブジェクト定義のサポートは削除されません。

要求ビジネス・オブジェクトの受け入れ

ビジネス・オブジェクトを要求として送信すると、そのビジネス・オブジェクトは、トランザクションに障害が発生しない限り、インターフェース内の宛先コネクターをエミュレートしているすべての Test Connector インスタンスの「ビジネス・オブジェクト要求リスト」ペインに表示されます。

受け入れた要求ビジネス・オブジェクトは、216 ページの『応答ビジネス・オブジェクトの編集』に示した方法で、必要に応じて編集できます。

応答ビジネス・オブジェクトの処理

応答ビジネス・オブジェクトは、インターフェース内のビジネス・オブジェクト要求の宛先のコネクターを Test Connector がエミュレートしている際に、Visual Test Connector から送信されるビジネス・オブジェクトです。要求ビジネス・オブジェクトの処理では、ビジネス・オブジェクト・インスタンス内の値を編集し、応答をブローカーに返送します。

応答ビジネス・オブジェクトの編集

Test Connector の宛先インスタンスでビジネス・オブジェクト要求を受信すると、通常、属性の値を編集する必要が生じます。例えば、関係に関与している基本キー属性への固有値の設定や、ビジネス・オブジェクトの実際の値に応じて異なった応答をするマップやコラボレーション・ロジックをテストするために、その他の属性の値の変更を行なうことが必要となります。ビジネス・オブジェクト属性の値を設定するには、以下の手順を実行します。

1. 「ビジネス・オブジェクト要求リスト」ペインで、ビジネス・オブジェクト・インスタンスを選択する。
2. メニュー・バーから、「**要求**」>「**応答を編集**」を選択する。
3. 以下の方法で、ビジネス・オブジェクトの属性を編集する。
   - 213 ページの『ビジネス・オブジェクト属性の値の設定』に示した方法の 1 つを使用して、ビジネス・オブジェクト属性の値を変更する。
   - ビジネス・オブジェクト属性の値を、ビジネス・オブジェクト定義で指定されたデフォルト値に設定するには、「**ビジネス・オブジェクトをデフォルトにリセット**」をクリックする。
   - ビジネス・オブジェクトのすべての属性の値をクリアするには、「**ビジネス・オブジェクト値をクリア**」をクリックする。
   - ビジネス・オブジェクトの属性に、ファイルからテスト・データを取り込むには、「**ビジネス・オブジェクトをロード**」をクリックする。

     保管されているデータをビジネス・オブジェクト要求にロードする機能は、応答として送信する前に応答ビジネス・オブジェクトにデータを取り込む必要がある場合に非常に役立ちます。応答データを必要とする各属性のために値を手動で入力する代わりに、値を 1 度だけ入力し、ビジネス・オブジェクトを保管し (214 ページの『ビジネス・オブジェクトの保管』に示す方法で)、保管されたデータを以降のテスト時にロードすることができます。

応答ビジネス・オブジェクトの送信

要求ビジネス・オブジェクトを受け付けたら、必要に応じて編集し、応答として返送します。

217 ページの表 26 に、Test Connector の応答オプションと、それに対応する C++ コネクターと Java コネクターの双方のコネクター戻りコードを示します。C++ または Java コネクターの戻りコードについての詳細は、「*コネクター開発ガイド (Java 用または C++ 用)*」を参照してください。

表 26. *Test Connector* の応答タイプとコネクター戻りコード

Test Connector の応答タイプ	**C++ コネクターの戻りコード**	**Java コネクターの戻りコード**
成功	BON_SUCCESS	SUCCESS
失敗	BON_FAIL	FAIL
複数ヒット	BON_MULTIPLE_HITS	MULTIPLE_HITS
コンテンツ別検索の失敗	BON_FAIL_RETRIEVE_BY_CONTENT	RETRIEVEBYCONTENT_FAILED
未検出	BON_BO_DOES_NOT_EXIST	BO_DOES_NOT_EXIST
値の重複	BON_VALDUPES	VALDUPES

要求ビジネス・オブジェクトに応答する手順は次のとおりです。

1. 「ビジネス・オブジェクト要求リスト」ペインで、ビジネス・オブジェクトを選択する。
2. メニュー・バーから、「**要求**」>「**応答**」を選択する。
3. 「**応答**」サブメニューから、項目を選択する。

ビジネス・オブジェクト・インスタンスの比較

Test Connector では、同じタイプの 2 つのビジネス・オブジェクトを比較して、双方の値が異なっている属性を表示できます。この機能を使用すると、トランザクション実行中の異なる時点での、ビジネス・オブジェクトの変化を確認できます (例えば、同一のビジネス・オブジェクトについて、統合ブローカーに送信された時点のものと、統合ブローカーによって更新された後のものとを比較できます)。 2 つのビジネス・オブジェクトを比較する手順は次のとおりです。

1. 209 ページの『要求ビジネス・オブジェクトの作成』または 215 ページの『ビジネス・オブジェクトのロード』の説明に従って、要求ビジネス・オブジェクトのインスタンスを作成する。
2. 要求ビジネス・オブジェクトのインスタンスと比較する応答ビジネス・オブジェクトのインスタンスを、「ビジネス・オブジェクト要求リスト」ペインで選択する。
3. メニュー・バーから、「**編集**」>「**ビジネス・オブジェクトを比較**」を選択する。

   Test Connector で「ビジネス・オブジェクトの比較 (Compare Business Objects)」ウィンドウが開き、2 つのビジネス・オブジェクトで値の異なる属性を示したテーブルが表示されます。218 ページの図 45 に、2 つのビジネス・オブジェクト・インスタンスの比較を示します。

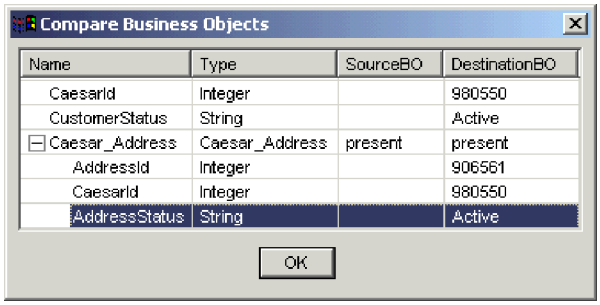


図 45. 「ビジネス・オブジェクトの比較」ウィンドウ

4. 「**OK**」をクリックして、ウィンドウを閉じる。

付録 G. WebSphere Business Integration Adapters のアップグレード

この付録では、WebSphere Business Integration Adapter Framework およびアダプターの新規リリースへのアップグレード処理について説明します。本章は、以下のトピックから構成されています。

- 『前提事項』
- 『WebSphere Business Integration Adapters のインストール』
- 220 ページの『WebSphere Application Server 用 Service Pack のインストール』
- 220 ページの『既存のアダプターのアップグレード』

前提事項

このアップグレード手順では、以下の条件であることが想定されています。

- WebSphere Application Server を統合ブローカーとして使用しているシステムをアップグレードする。
- ご使用のシステムに、現在、WebSphere Business Integration Adapters バージョン 2.3.1 がインストールされている。
- 必要なファイルはすべてバックアップしている。
- 実行中のシステムおよびすべてのコネクターが停止している。
- リリース情報を読み、インストールに影響する可能性があるリリース固有のアップグレード情報を理解している。オンラインのリリース情報は、WebSphere Business Integration Adapters InfoCenter (http://www.ibm.com/websphere/integration/wbiadapters/infocenter) で参照できます。
- 開発環境でアップグレードし、テストが完了してから実稼働環境のアップグレードに移行する。

WebSphere Business Integration Adapters のインストール

WebSphere Business Integration Adapters をインストールするには、「*WebSphere Business Integration Adapters インストール・ガイド*」に記載されている説明に従ってください。この資料は、Web アドレス http://www.ibm.com/software/websphere/wbiadapters/infocenter で入手できます。

インストール対象の WebSphere Business Integration Adapter コンポーネントを指定するように求めるプロンプトがインストーラー画面に表示されたら、次のものを選択します。

- WebSphere Business Integration Adapter Framework
- WebSphere Business Integration Adapter Development Kit
- アップグレードするすべてのアダプター
- インストールするすべての新規アダプター

WebSphere Application Server 用 Service Pack のインストール

WebSphere Business Integration Adapters バージョン 2.3.1 からアップグレードする場合、WebSphere Application Server Enterprise Edition 5.0 を稼働している必要があります。アップグレードするには、WebSphere Application Server Enterprise Edition 5.0 Fix Pack 2 を適用してください。この製品は以下の Web サイトから入手できます。

http://www.ibm.com/support/docview.wss?rs=180&tc=SSEQTP&uid=swg24005055

WebSphere MQ 用 Service Pack のインストール

WebSphere Business Integration Adapters バージョン 2.3.1 からアップグレードする場合、WebSphere MQ CSD05 Service Pack を適用してください。このパッケージは以下の URL から入手できます。

http://www.ibm.com/software/integration/mqfamily/support/summary/

既存のアダプターのアップグレード

以上が WebSphere Business Integration Adapters バージョン 2.3.1 の既存アダプターのアップグレードに必要なステップです。

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。

System Manager およびその他のパースペクティブには、Eclipse Project

(http://www.eclipse.org/) により開発されたソフトウェアが含まれています。

IBM 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