BEA WebLogic Integration™

WebLogic Integration 移行ガイド

リリース 7.0
マニュアルの日付 : 2002 年 6 月

著作権

限定的権利条項

商標または登録商標

BEA WebLogic Integration 移行ガイド

パート番号	日付	ソフトウェアのバージョン
なし	2002 年 6 月	7.0

目次

3. 移行に関するその他のトピック

このマニュアルの内容

このマニュアルの内容（『*BEA WebLogic Integration 移行ガイド*』）は以下のとおりです。

- 第 1 章「移行の概要」では、WebLogic Integration 7.0 に移行する際に役立つバックグラウンド情報について説明します。
- 第 2 章「WebLogic Integration 2.1 から WebLogic Integration 7.0 への移行」では、BEA WebLogic Integration 2.1 または BEA WebLogic Integration 2.1 サービス パック 1 (SP1) から BEA WebLogic Integration 7.0 に移行するための手順について説明します。
- 第 3 章「移行に関するその他のトピック」では、WebLogic Server の信頼関係およびクラスタ化された WebLogic Integration アプリケーションの移行について説明します。また、Application Integration アダプタ EAR ファイルを構成する手順についても説明します。

対象読者

『*BEA WebLogic Integration 移行ガイド*』は、WebLogic 7.0 に移行する予定の WebLogic Integration ユーザ用に設計されています。

e-docs Web サイト

BEA 製品のドキュメントは、BEA Systems, Inc. の Web サイトで入手できます。BEA のホーム ページで [製品のドキュメント] をクリックするか、または「e-docs」という製品ドキュメント ページ（`http://edocs.beasys.co.jp/e-docs/index.html`）を直接表示してください。

このマニュアルの印刷方法

Web ブラウザの [ファイル | 印刷] オプションを使用すると、Web ブラウザからこのマニュアルを一度に 1 ファイルずつ印刷できます。

このマニュアルの PDF 版は、Web サイトで入手できます。WebLogic IntegrationPDF を Adobe Acrobat Reader で開くと、マニュアルの全体（または一部分）を書籍の形式で印刷できます。PDF を表示するには、WebLogic Integration ドキュメントのホーム ページを開き、[PDF 版] ボタンをクリックして、印刷するマニュアルを選択します。

Adobe Acrobat Reader がない場合は、Adobe の Web サイト（`http://www.adobe.co.jp/`）で無料で入手できます。

関連情報

以下の関連情報があります。

- BEA WebLogic Server 7.0 ドキュメント（`http://edocs.beasys.co.jp/e-docs/wls/docs70/index.htm`）
- BEA WebLogic Integration 2.1 ドキュメント (`http://edocs.beasys.co.jp/e-docs/wlintegration/v2_1/index.html`)
- BEA WebLogic Integration 7.0 ドキュメント (http://e-docs.beasys.co.jp/wli/docs70/index.html)

サポート情報

WebLogic Integration のドキュメントに関するユーザからのフィードバックは弊社にとって非常に重要です。質問や意見などがあれば、電子メールで **docsupport-jp@bea.com** までお送りください。寄せられた意見については、WebLogic Integration のドキュメントを作成および改訂する BEA の専門の担当者が直に目を通します。

電子メールのメッセージには、WebLogic Integration 7.0 リリースのドキュメントをご使用の旨をお書き添えください。

本バージョンの BEA WebLogic Integration について不明な点がある場合、または BEA WebLogic Integration のインストールおよび動作に問題がある場合は、BEA WebSUPPORT (**websupport.bea.com/custsupp**) を通じて BEA カスタマ サポートまでお問い合わせください。カスタマ サポートへの連絡方法については、製品パッケージに同梱されているカスタマ サポート カードにも記載されています。

カスタマ サポートでは以下の情報をお尋ねしますので、お問い合わせの際はあらかじめご用意ください。

- お名前、電子メール アドレス、電話番号およびファックス番号
- 会社の名前と住所
- お使いの機種とコード番号
- 製品の名前とバージョン
- 問題の状況と表示されるエラー メッセージの内容

表記規則

このマニュアルでは、全体を通して以下の表記規則が使用されています。

表記法	適用
〔Ctrl〕+〔Tab〕	複数のキーを同時に押すことを示す。
斜体	強調または書籍のタイトルを示す。

等幅テキスト	コード サンプル、コマンドとそのオプション、データ構造体とそのメンバー、データ型、ディレクトリ、およびファイル名とその拡張子を示す。等幅テキストはキーボードから入力するテキストも示す。
	例
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

太字の等幅テキスト	コード内の重要な箇所を示す。 *例* `void **commit** ( )`
斜体の等幅テキスト	コード内の変数を示す。 *例* `String *expr*`
すべて大文字のテキスト	デバイス名、環境変数、および論理演算子を示す。 *例* LPT1 SIGNON OR

表記法	適用
{ }	構文の中で複数の選択肢を示す。実際には、この括弧は入力しない。

[ ]	構文の中で任意指定の項目を示す。実際には、この括弧は入力しない。
	例
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

|	構文の中で相互に排他的な選択肢を区切る。実際には、この記号は入力しない。

...	コマンドラインで以下のいずれかを示す。
	■ 引数を複数回繰り返すことができる
	■ 任意指定の引数が省略されている
	■ パラメータや値などの情報を追加入力できる
	実際には、この省略記号は入力しない。
	例
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

. . .	コード サンプルまたは構文で項目が省略されていることを示す。 実際には、この省略記号は入力しない。

1　移行の概要

このマニュアルでは、WebLogic Integration 2.1 または WebLogic Integration 2.1 サービス パック 1 (SP1) から WebLogic Integration 7.0 に移行するために必要な手順について説明します。

この章の内容は以下のとおりです。

- 移行する理由
- WebLogic Integration 2.1 以前のリリースからの移行
- WebLogic Integration の WebLogic Server または WebLogic Portal アプリケーションへの追加

移行する理由

WebLogic Integration 7.0 と WebLogic Integration 2.1 または WebLogic Integration 2.1 サービス パック 1 (SP1) では次の点が異なるため、移行が必要となります。

- WebLogic Integration 7.0 の新規のデータベース スキーマ
- RosettaNet ワークフローの変更
- セキュリティ キーストアの変更
- クラスタ化の変更
- WebLogic Server リリース 6.1 および 7.0 間での変更点
  - 起動スクリプト
  - ディレクトリ構造
  - Java メッセージング サービス
  - エンタープライズ JavaBean (Enterprise JavaBeans: EJB) 2.0
  - サーブレット
  - スレッド プールのサイズ
  - Web アプリケーション
  - WebLogic Server クラスタ
  - Apache Xerces XML パーサ
  - Apache Xalan XML トランスフォーマ
  - セキュリティ

注意： WebLogic Integration 7.0 はソフトウェア フレームワークであり、WebLogic Server 7.0 の上に構築されたサービス セットであるため、これらの違いは、WebLogic Integration 7.0 への移行時に重要となります。

WebLogic Server 7.0 の変更点の詳細については、『*BEA WebLogic Server 7.0 へのアップグレード*』の「WebLogic Server 6.x からバージョン 7.0 へのアップグレード」を参照してください。このマニュアルは、BEA WebLogic Server マニュアル セットの次の URL にあります。

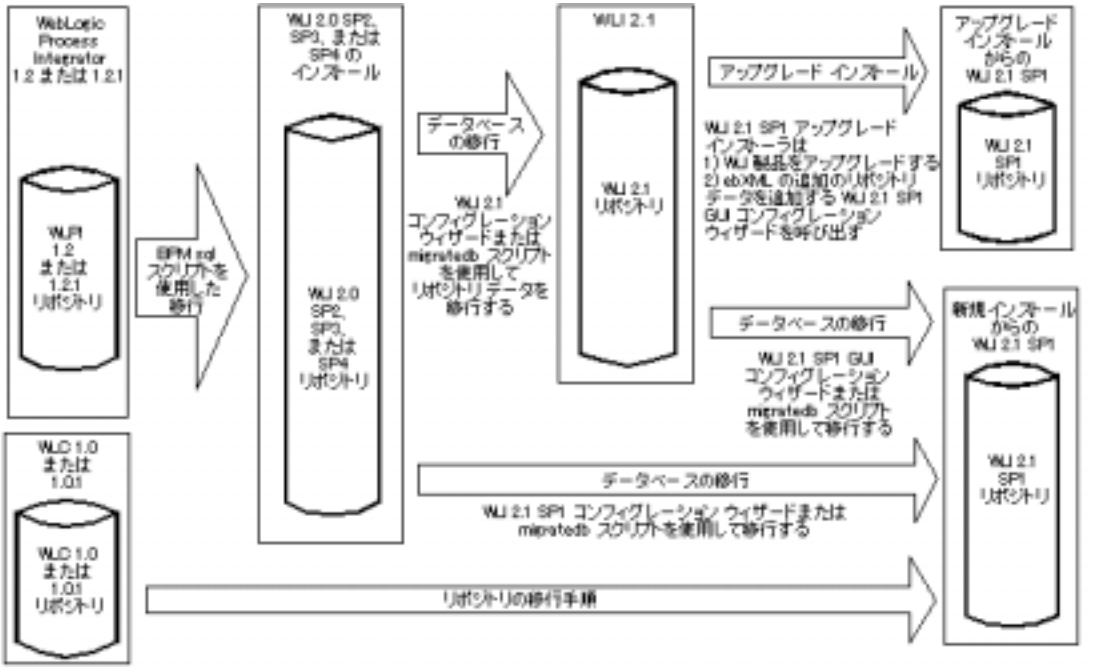
```
http://edocs.beasys.co.jp/e-docs/wls/docs70/upgrade/index.html
```

注意： このマニュアルで使用されている*アップグレード*および*移行*は同義語です。*移行*とは、従来のリリースから新規のリリースへのアップグレードを指します。このアップグレード プロセスを WebLogic Server の 1 つのインスタンスから別のものへのクラスタ対応サービスの移動と混同しないでください。

WebLogic Integration 2.1 以前のリリースからの移行

WebLogic Integration 2.1 サービス パック 1 (SP1) 以前のリリースから WebLogic Integration 7.0 に移行する場合、次の図のようにまず WebLogic Integration 2.1 SP1 へアップデートする必要があります。

図 1-1 WebLogic Integration 2.1 サービス パック 1 への移行パス



WebLogic Integration 2.1 SP1 からの移行後、このガイドの WebLogic Integration 2.1 SP1 から WebLogic Integration 7.0 への移行のための手順に従います。

注意： We b Logic Integration 2.0 から WebLogic Integration 7.0 へ移行する場合、次の 2 段階の手順を実行して移行することをお勧めします。1) WebLogic Integration 2.0 から WebLogic Integration 2.1 SP1 へ移行し、2) WebLogic Integration 2.1 SP1 から WebLogic Integration 7.0 へ移行します。移行手順を最小限に押さえられるため、この移行手順をお勧めします。次の 3 段階の手順の代わりにこの 2 段階の手順をお勧めします。1) WebLogic Integration 2.0 から WebLogic Integration 2.1 へ移行し、2) WebLogic Integration 2.1 から WebLogic Integration 2.1 SP1 へ移行し、3) WebLogic Integration 2.1 SP1 から WebLogic Integration 7.0 へ移行します。

以前のリリースから WebLogic Integration 2.1 SP1 への移行の詳細については、次の URL の『*WebLogic Integration リリース 2.1 への移行*』を参照してください。

```
http://edocs.beasys.co.jp/e-docs/wlintegration/v2_1/migrate/index
.htm
```

WebLogic Integration の WebLogic Server または WebLogic Portal アプリケーションへの追加

第 2 章「WebLogic Integration 2.1 から WebLogic Integration 7.0 への移行」に記載されている手順は、WebLogic Integration 2.1 または WebLogic Integration 2.1 SP 1 から WebLogic Integration 7.0 への手順です。WebLogic Integration を WebLogic Server または WebLogic Portal の以前のリリースの 7.0 バージョンに基づいた既存のアプリケーションに追加する方法については説明されていません。WebLogic Integration を WebLogic Server または WebLogic Portal アプリケーションに追加する場合、追加に影響を及ぼす可能性のあるセキュリティ レルムの違いが存在します。詳細については、次の URL の BEA WebLogic Platform ドキュメント群に含まれている『*WebLogic Platform 7.0 Security の概要*』を参照してください。

```
http://edocs.beasys.co.jp/e-docs/platform/docs70/secintro/index
.html
```

2 WebLogic Integration 2.1 から WebLogic Integration 7.0 への移行

この章では、BEA WebLogic Integration 2.1 または BEA WebLogic Integration 2.1 サービス パック 1 (SP1) から BEA WebLogic Integration 7.0 に移行する手順について説明します。

この章の内容は以下のとおりです。

- 手順 1. WebLogic Integration 2.1 データベース接続情報の取得
- 手順 2. WebLogic Integration 2.1 アプリケーションの停止
- 手順 3. アプリケーションのバックアップ
- 手順 4. WebLogic Integration 7.0 のインストール
- 手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション
- 手順 6. データベースの移行
- 手順 7. WebLogic Integration アプリケーションのコンポーネントの移行
- 手順 8. セキュリティ レルム データの移行
- 手順 9. セキュリティのキーストアへの移行
- 手順 10. RosettaNet ワークフローの移行
- 手順 11. アプリケーション ビューのデプロイ
- 手順 12. WebLogic Integration アプリケーションの開始およびテスト

手順 1. WebLogic Integration 2.1 データベース接続情報の取得

この手順の後半に（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」）WebLogic Integration 2.1 または WebLogic Integraiton 2.1 SP1 のインストールのデータベース コンフィグレーションについての情報が必要となります。その情報を今取得して、後で使用できるようにします。

情報を取得するには、次の手順を実行します。

1. WebLogic Integration 2.1 または WebLogic Integration 2.1 SP 1 データベース ウィザードを開始するために、プラットフォームに合わせて適切な手順を実行します。
   - Windows:
   a. WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 データベース情報を含むドメインを選択します。
   b. 該当するドメインのデータベース ウィザードを開始します。

対象ドメイン	選択
`wlidomain`	［スタート｜ BEA WebLogic E-Business Platform ｜ WebLogic Integration 2.1 ｜ Configure］
`samples`	［スタート｜ BEA WebLogic E-Business Platform ｜ WebLogic Integration 2.1 ｜ Samples ｜ Configure］
`eaidomain`	［スタート｜ BEA WebLogic E-Business Platform ｜ WebLogic Integration 2.1 ｜ Additional Preconfigured Domains ｜ EAI Domain ｜ Configure］
`bpmdomain`	［スタート｜ BEA WebLogic E-Business Platform ｜ WebLogic Integration 2.1 ｜ Additional Preconfigured Domains ｜ BPM Domain ｜ Configure］

[Choose Configuration Option] ダイアログ ボックスが表示されます。

- UNIX:

a. 次のコマンドを実行します。

```
cd WLI_HOME/bin
wliconfig
```

[Choose BEA Home Directory] ダイアログ ボックスが表示されます。

b. 既存の BEA ホーム ディレクトリを選択し、[Next] をクリックします。

[Choose Domain to Configure] ダイアログ ボックスが表示されます。

c. ドメインを選択し、[Next] をクリックします。

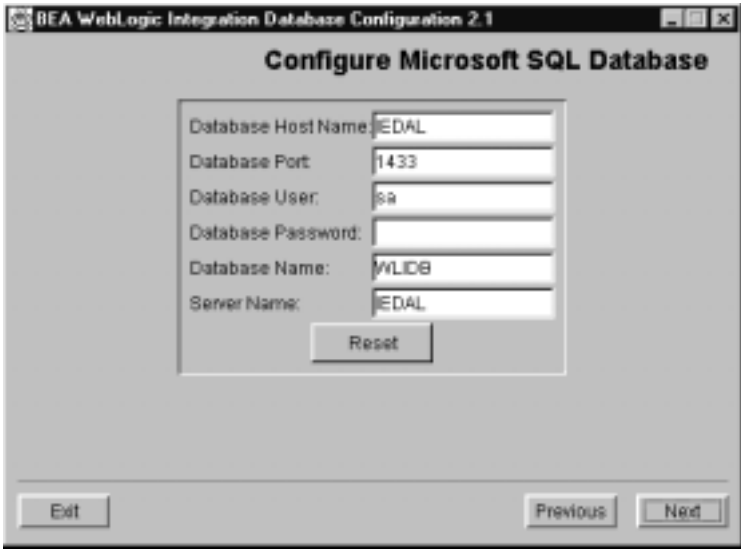
[Choose Configuration Option] ダイアログ ボックスが表示されます。

2. [Switch Database] を選択し、[Next] をクリックします。

[Select Database] ダイアログ ボックスが表示されます。

3. データベースの種類（Oracle、Microsoft SQL Server、または Sybase）を選択し、[Next] を選択します。

［Configure *database_type* Database］ダイアログ ボックスが表示されます。ウィンドウ タイトルの *database_type* が選択したデータベースの種類の名前に置きかえられます。この例では Microsoft SQL が選択されているため、［Configure Microsoft SQL Database］ダイアログ ボックスが表示されます。



4. フィールドに含まれるデータベース コンフィグレーション情報を記録します。この情報は 2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で必要となります。
5. ［Exit］をクリックします。

手順 2. WebLogic Integration 2.1 アプリケーションの停止

WebLogic Integration 7.0 に移行する前に WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 アプリケーションを停止します。次のことを確認します。

- WebLogic Integration アプリケーションが静止状態にあること。
- アプリケーションの WebLogic Server のすべてのインスタンスがシャットダウンされていること。
- メッセージが送信されていないこと。

手順 3. アプリケーションのバックアップ

WebLogic Integration 7.0 に移行する前に次のバックアップを行うことをお勧めします。

- データベース全体
- WebLogic Integration アプリケーションを含む次のディレクトリおよびファイル
  - `config.xml` ファイル
  - 証明書およびプライベート キー（使用している場合）
  - ビジネス オペレーションで使用されているクラスおよび JAR ファイル（使用している場合）
- セキュリティ レルム データ

すべての WebLogic Integration リポジトリ情報およびワークフローをエクスポートすることもお勧めします。

- WebLogic Integration リポジトリ情報のエクスポート方法については、『*B2B Integration Administration Console オンライン ヘルプ*』の「B2B Integration のコンフィグレーション」の「リポジトリ データのエクスポート」を参照してください。エクスポートのリポジトリ エンティティを選択する場合、[エクスポートする範囲] フィールドの [All] を選択します。
- WebLogic Integration Studio を通して WebLogic Integration ワークフローをエクスポートする方法については、『*WebLogic Integration Studio ユーザーズ ガイド*』の「ワークフロー パッケージのインポートとエクスポート」を参照してください。

手順 4. WebLogic Integration 7.0 のインストール

2 つのモードのいずれかで WebLogic Integration 7.0 をインストールします。

- カスタム インストール ― このモードでインストールし、WebLogic Integration コンポーネントを選択した場合、WebLogic Server コンポーネントも自動的にインストールされます。
- 標準的なインストール ― このモードでインストールした場合、WebLogic Server、WebLogic Integration、WebLogic Portal、および WebLogic Workshop コンポーネントがデフォルトで自動的にインストールされます。

手順については次の URL の BEA WebLogic Platform マニュアル セットに含まれている『*BEA WebLogic Platform インストール ガイド*』を参照してください。

```
http://edocs.beasys.co.jp/e-docs/platform/docs70/install/index.html
```

警告： インストール完了後に新しい WebLogic Integration リポジトリまたはデータベースを作成しないでください。`RunSamples` スクリプトを実行しないでください。

手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション

WebLogic Server ドメインのディレクトリ構造はリリース 6.x と 7.0 では異なります。この変更は、環境変数の多くのスクリプトおよび設定に影響します。たとえばディレクトリ構造の変更はクラスパスおよびパス環境変数および WebLogic Integration を起動するスクリプト（`startWebLogic`）の設定に影響します。そのため、この手順は 2-14 ページの「手順 7. WebLogic Integration アプリケーションのコンポーネントの移行」に記載のとおり、新しい WebLogic Integration 7.0 ドメインの作成および WebLogic Integration アプリケーション特有のコンポーネントのドメインへの移動について説明しています。

次の手順では、1 つのサーバ（スタンドアロン）のコンフィグレーションの新しいドメインを作成できます。複数のサーバのコンフィグレーションの作成方法については『*WebLogic Integration ソリューションのデプロイメント*』の「クラスタ デプロイメントのコンフィグレーション」の「手順 2. WebLogic Integration ドメインの作成」を参照してください。

1 つのサーバの新しいドメインを作成およびカスタマイズするには、次の手順を実行します。

1. コンフィグレーション ウィザードを実行するために、プラットフォームに適した手順を実行します。

- Windows システムの場合、3 種類の方法が存在します。
  - メニューからコンフィグレーション ウィザードを開始するには、次の手順を実行します。

    [スタート | プログラム | BEA WebLogic Platform 7.0 | Domain Configuration Wizard] を選択します。
  - コマンドラインからグラフィカル モードのコンフィグレーション ウィザードを開始するには、次の手順を実行します。

    ```
    cd c:\bea\weblogic700\integration
    setenv.cmd
    cd c:\bea\weblogic700\common\bin
    dmwiz
    ```
  - コンソールまたはテキスト ベース モードのコンフィグレーション ウィザードを開始するには、

    ```
    cd c:\bea\weblogic700\integration
    setenv.cmd
    cd c:\bea\weblogic700\common\bin
    dmwiz console
    ```
- UNIX システムの場合、2 種類の方法が存在します。
  - グラフィカル モードからコンフィグレーション ウィザードを開始するには、次の手順を実行します。

    ```
    cd /home/joe/bea/weblogic700/integration
    . setenv.sh
    cd /home/joe/bea/weblogic700/common/bin
    dmwiz.sh
    ```
  - コンソールまたはテキスト ベース モードのコンフィグレーション ウィザードを開始するには、次の手順を実行します。

    ```
    cd /home/joe/bea/weblogic700/integration
    . setenv.sh
    cd /home/joe/bea/weblogic700/common/bin
    dmwiz.sh console
    ```

[ドメインのタイプと名前を選択] ウィンドウが表示されます。

詳細については、次の URL の BEA WebLogic Platform マニュアル セットに含まれている『*Configuration Wizard の使い方*』を参照してください。

```
http://edocs.beasys.co.jp/e-docs/platform/docs70/confgwiz/index.html
```

2. アプリケーションで必要な WebLogic Integration のコンポーネントが含まれるドメイン テンプレートを選択します。

WebLogic Integration アプリケーションが次の WebLogic Integration 機能を使用している場合	次のドメイン テンプレートを選択します。
Application Integration、Data Integration、BPM (Business Process Management)、および B2B Integration	WLI (WebLogic Integration) ドメイン
Application Integration、Data Integration、および BPM (Business Process Management)	EAI (Enterprise Application Integration) ドメイン
BPM (Business Process Management)	BPM (Business Process Management) ドメイン

これらのテンプレートの詳細については、『*WebLogic Integration の起動、停止およびカスタマイズ*』の「はじめに」の「WebLogic Integration コンフィグレーション テンプレート」および次の URL の BEA WebLogic Platform マニュアル セットにある『*Configuration Wizard Template リファレンス*』を参照してください。

```
http://edocs.beasys.co.jp/e-docs/platform/docs70/template/index.html
```

警告： 新しいドメインを作成するための WebLogic Integration テンプレートを選択してください。WebLogic Server または WebLogic Portal テンプレートは使用しないでください。WebLogic Integration テンプレートを指定することにより、互換モードの WebLogic Server 6.x セキュリティ レルムに基づいたドメインがこの手順で作成されます。この WebLogic Integration のリリースでは、LDAP に基づいた新しい WebLogic Server 7.0 レルムはサポートされません。WebLogic Server テンプレートを選択して新しいドメインを作成した場合、新しいドメインは LDAP に基づいた新しい WebLogic Server 7.0 セキュリティ レルムを使用します。

注意： 使用している WebLogic Integration 2.1 アプリケーションで使用するものと同じドメイン タイプを選択することをお勧めします。たとえば、WebLogic Integration 2.1 アプリケーションでコンフィグレーション済みの EAI ドメイン を使用した場合、この手順の EAI ドメイン テンプレートを選択します。

3. [サーバ タイプを選択] ダイアログ ボックスから [Single Server (Standalone Server)] を選択します。
4. 必要に応じて新しいドメインを作成するために必要な情報を入力します。このプロセスが完了したときに、[コンフィグレーション ウィザードが完了しました] ダイアログ ボックスから [コンフィグレーション ウィザードを終了します] を選択し、[Done] ボタンをクリックして、コンフィグレーション ウィザードを終了します。
5. （省略可能）コメントに組み込まれたコンフィグレーション情報を含む、生成された `config.xml` ファイルのコピーを保存します。WebLogic Server のインスタンスが開始されたときにコメントは削除されます。使用しているオペレーティング システムに適したコマンドを入力して既存の `config.xml` ファイルのコピーを作成します。
   - Windows:

     ```
     copy config.xml config.xml.backup
     ```

   - UNIX:

     ```
     cp config.xml config.xml.backup
     ```

6. 新しいドメインのデータベース接続情報のコンフィグレーションを行います。手順 1 から 4 で作成した新しいドメインで WebLogic Integration Database Wizard (`wliconfig`) を起動します。

   たとえば、デフォルトのロケーションで `mydomain` という名前のドメインを作成した場合、使用しているオペレーティング システムに合ったコマンドを入力します。
   - Windows:

     ```
     cd %BEA_HOME%\user_projects\mydomain
     wliconfig
     ```

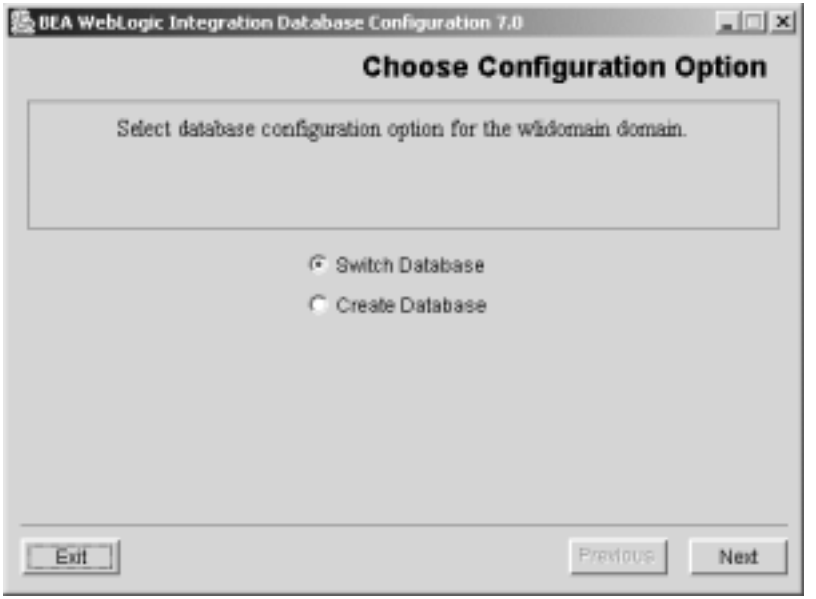
   - UNIX:

     ```
     cd $BEA_HOME/user_projects/mydomain
     wliconfig
     ```

   [コンフィグレーション オプションの選択] ウィンドウが表示されます。

7. 図のように、[データベースの切り替え] オプションを選択し、[Next] をクリックします。

図 2-1 [コンフィグレーション オプションの選択]



[データベースの選択] ウィンドウが表示されます。

8. WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 アプリケーションで使用しているものと同じデータベース タイプを選択します。

   [データベースのコンフィグレーション] ウィンドウが表示されます。

9. 2-2 ページの「手順 1. WebLogic Integration 2.1 データベース接続情報の取得」で記載したデータベース接続情報を入力します。[Next] をクリックします。

   データベース ウィザードのコンフィグレーションが終了したときに、[変更の成功] ウィンドウが表示されます。

10. [Finish] をクリックします。

データベース ウィザードの実行の詳細については『*WebLogic Integration の起動、停止およびカスタマイズ*』の「WebLogic Integration のカスタマイズ」の「データベース ウィザードの使用法」を参照してください。

警告： このドメインの WebLogic Server のインスタンスの開始およびデータベースの作成は行わないでください。

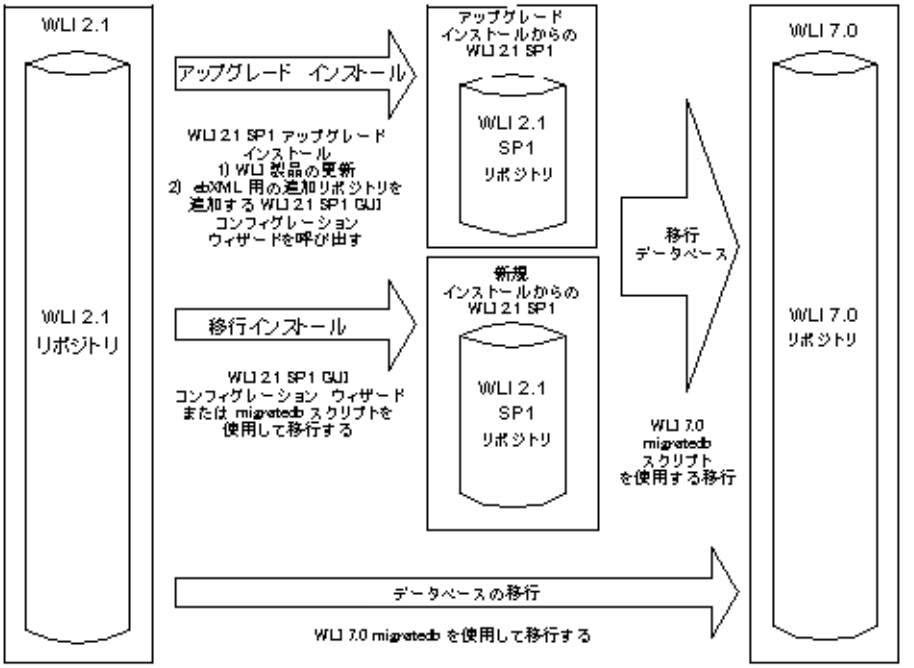
WebLogic Integration のシステム ID

WebLogic Integration 2.1 および WebLogic Integration 2.1 SP1 では 2 つのユーザ ID が使用されます。`wlpisystem` および `wlcsystem` です。WebLogic Integration 7.0 では 2 つのシステム ユーザ ID が 1 つのシステム ユーザ ID に置き換えられます。`wlisystem`. (『*WebLogic Integration の起動、停止およびカスタマイズ*』の「WebLogic Integration のカスタマイズ」の「新しいドメインの作成とカスタマイズ」に説明されているように) WebLogic Integration の新しいドメインを作成した場合、システム ユーザ ID、`wlisystem` のみが作成されます。

手順 6. データベースの移行

この節では WebLogic Integration 2.1 または WebLogic Integration 2.1 サービス パック 1 (SP1) から WebLogic Integration 7.0 フォーマットへのデータベース スキーマの変換手順について説明します。このアップグレード手順は、図 2-2 のデータベースの移行と記された灰色の矢印によって表されています。

図 2-2 WebLogic Integration 7.0 データベースの移行



警告： この移行の手順は 1 つのリポジトリをアップデートします。既存のデータベース インスタンスから新しいインスタンスへのリポジトリ データの移行はサポートされていません。

既存のリポジトリ データを WebLogic Integration 7.0 フォーマットにアップデートするには、次の手順を実行します。

1. WebLogic Integration ホーム ディレクトリに行き、`setenv` スクリプトを実行して最上位レベルの WebLogic Integration 環境変数を設定します。これらのタスクを行うために実行するコマンドは、使用するプラットフォームによって異なります。

- Windows:

```
cd c:\bea\weblogic700\integration
setEnv.cmd
```

- UNIX:

```
cd /home/joe/bea/integration
. setenv.sh
```

2. テキスト エディタで、*WLI_HOME*\dbscripts\migrate\SystemRepData.xml ファイルを開き、次のリストの太字表示された行を追加します。

コード リスト 2-1 SystemRepData.xml

```
<?xml version="1.0"?>
<!DOCTYPE wlc SYSTEM "WLC.dtd">
<wlc
       system-password="wlisystem"
       ignore-wlc="true"
>
```

3. WebLogic Integration ホーム ディレクトリの bin ディレクトリに移動します。たとえば、
   - Windows:

     ```
     cd c:\bea\weblogic700\integration\bin
     ```

   - UNIX:

     ```
     cd /home/joe/bea/weblogic700/integration/bin
     ```

4. 2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成したドメインで setdomain スクリプトを実行します。たとえば、
   - Windows:

     ```
     setdomain c:\bea\user_projects\mydomain
     ```

   - UNIX:

     ```
     setdomain /home/joe/bea/user_projects/mydomain
     ```

5. 2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」でコンフィグレーションを行ったデータベースで switchdb スクリプトを実行します。たとえば、

   ```
   switchdb oracle
   ```

有効なオプションの詳細を含むこのコマンドの説明については『*WebLogic Integration の起動、停止およびカスタマイズ*』の「WebLogic Integration コマンド」を参照してください。

6. リポジトリ データを WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 から WebLogic Integration 7.0 ドメインにアップデートするための移行スクリプトを実行します。

   ```
   migratedb
   ```

手順 7. WebLogic Integration アプリケーションのコンポーネントの移行

2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」では新しい WebLogic Integration ドメインを作成しました。この手順では、WebLogic Server アプリケーション固有のコンポーネントを WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 アプリケーションから新しいドメインへ移行します。

注意： ここで使用されているように、*WebLogic Integration アプリケーション*とは、WebLogic Integration に基づき開発したアプリケーションを示します。`config.xml` ファイルの `WebLogic Application` アプリケーション要素とは異なります。

WebLogic Server アプリケーション固有のコンポーネントには次のようなものがあります。

- JMS キュー
- エンタープライズ アプリケーション ― エンタープライズ アプリケーションには複数の EJB、Web アプリケーション、および Connector Modules が含まれます。
- エンタープライズ JavaBean (Enterprise JavaBeans: EJB)
- Web アプリケーション
- コネクタ

警告： このアプリケーションのために開発したアプリケーション固有コンポーネントのみを移行します。WebLogic Integration が使用するコンポーネントは移行しないでください。たとえば、WebLogic Integration で使用する JMS キューは移行しませんが、WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 アプリケーションのために開発したカスタム JMS キューは移行する必要があります。We b Logic Integration で使用する JMS キューは、2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で新しいドメインを作成するときに生成される `config.xml` ファイルに自動的に含まれます。

警告： WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 ドメイン ディレクトリから WebLogic Integration 7.0 ドメイン ディレクトリへ、ファイル（`config.xml` など）をコピーしないでください。WebLogic Server 7.0 ドメインのディレクトリ構造の変更のため、WebLogic Server 6.x ファイルは WebLogic Server 7.0 と互換性がありません。

これらの WebLogic Server コンポーネントの移行手順および WebLogic Server 6.x から WebLogic Server 7.0 への移行の詳細については、『*BEA WebLogic Server 7.0 へのアップグレード*』の「WebLogic Server 6.x からバージョン 7.0 へのアップグレード」を参照してください。このマニュアルは、BEA WebLogic Server マニュアル セットの次の URL にあります。

```
http://edocs.beasys.co.jp/e-docs/wls/docs70/upgrade/index.html
```

注意： このマニュアルで使用されているように*アップグレード*および*移行*は同義語です。*移行*とは、従来のソフトウェアのリリースから新しいリリースへのアップグレードを指します。このアップグレード プロセスを WebLogic Server の 1 つからのクラスタ対応サービスの移動と混同しないでください。

WebLogic Integration 7.0 は WebLogic Server 7.0 の上に構築されているため、WebLogic Server 6.x から WebLogic Server 7.0 への移行は、WebLogic Integration 7.0 の移行に影響します。

警告： WebLogic Server 7.0 は LDAP に基づく新しいセキュリティ レルムをサポートしています。WebLogic Integration 7.0 は LDAP に基づく新しいセキュリティ レルムはサポートしていません。WebLogic Integration 7.0 は File および RDBMS レルムの両方をサポートしている Compatibility レルムをサポートしています。

アプリケーション統合

使用しているアプリケーションが、WebLogic Integration によって提供されているアプリケーション統合機能を呼び出す場合、Application Integration アダプタ EAR ファイルのコンフィグレーションを行う必要があります。手順については、3-3 ページの「Application Integration アダプタ EAR ファイルのコンフィグレーション」を参照してください。

アプリケーション固有の JAR ファイル

WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 アプリケーションのアプリケーション固有 JAR ファイルを作成する場合、*WLI_HOME*/lib ディレクトリ（*WLI_HOME* は WebLogic Integration 7.0 ホーム ディレクトリの場所を表す）にそれらの JAR ファイルをコピーする必要があります。

警告： アプリケーション統合 JAR ファイルを WebLogic Integration 2.1 または 2.1 SP1 アプリケーションから WebLogic Integration 7.0 アプリケーションへコピーしないでください。詳細については、2-18 ページの「アプリケーション統合 CLASSPATH および アダプタ パッケージを変更する」を参照してください。

アプリケーション固有の JAR のエントリを 2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した WebLogic Integration 7.0 ドメインの `config.xml` および `application.xml` ファイルに追加する必要があります。

WebLogic Integration J2EE コンポーネントは分解された形式でデプロイされます。分解された形式で J2EE コンポーネントをデプロイするには、`config.xml` ファイル内で JAR ファイルを指定するときに新しい構文を使用する必要があります。WebLogic Integration 2.1 または WebLogic Integration 2.1 `config.xml` ファイルから WebLogic Integration 7.0 `config.xml` ファイルにアプリケーション固有の JAR エントリを直接カット アンド ペーストしないでください。代わりに『*WebLogic Integration の起動、停止およびカスタマイズ*』の「WebLogic Integration のカスタマイズ」の「WebLogic Integration Application 要素に EJB を追加する」の手順に従ってください。

startWebLogic スクリプト

WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 アプリケーションで実行する `startWebLogic` スクリプトにカスタム変更を加えた場合、2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した WebLogic Integration 7.0 の `startWebLogic` スクリプトに同じ変更を加える必要があります。たとえば、ビジネス オペレーションを含んだアプリケーション固有 JAR ファイルを `startWebLogic` スクリプトの `CLASSPATH` に追加した場合、同じ JAR ファイルを WebLogic Integration 7.0 `startWebLogic` スクリプト（Windows システムの場合は `startWebLogic.cmd`、UNIX システムの場合は `startWebLogic.sh`）の `CLASSPATH` にも追加する必要があります。

詳細については、『*WebLogic Integration の起動、停止およびカスタマイズ*』の「WebLogic Integration のカスタマイズ」にある「Java Class の CLASSPATH への追加」を参照してください。

setenv スクリプト

WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 アプリケーションの `setenv` スクリプトがカスタマイズされている場合、2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した WebLogic Integration 7.0 ドメインの `setenv` スクリプトを同様にカスタマイズする必要があります。たとえばビジネス オペレーションを含むアプリケーション固有の JAR ファイルを `setenv` スクリプトの `CLASSPATH` に追加した場合、同じ JAR ファイルを WebLogic Integration 7.0 `setenv` スクリプトの `CLASSPATH` に追加する必要があります。このスクリプトの名前は、Windows システムでは `setenv.cmd` であり、UNIX システムでは `setenv.sh` です。

詳細については、『*WebLogic Integration の起動、停止およびカスタマイズ*』の「WebLogic Integration のカスタマイズ」にある「Java Class の CLASSPATH への追加」を参照してください。

アプリケーション統合 CLASSPATH および アダプタ パッケージを変更する

WebLogic Integration 2.1 または WebLogic Integration 2.1 SP 1 アプリケーションでは、WebLogic Server のインスタンスの システム CLASSPATH にアダプタ Java クラスを追加する必要があります。WebLogic Integration 7.0 アプリケーションでは、アダプタ Java クラスは 1 つの完全に独立した EAR ファイルにパッケージ化されている必要があります。アダプタ Java クラスまたは JAR ファイルを WebLogic Integration 7.0 インストレーションに移動しないでください。また、アダプタ クラスを WebLogic Integration CLASSPATH に追加しないでください。アダプタ EAR ファイルのコンフィグレーション手順については、3-3 ページの「Application Integration アダプタ EAR ファイルのコンフィグレーション」を参照してください。

B2B 転送サーブレット

WebLogic Integration 7.0 アプリケーションが WebLogic Integration で提供されている B2B (Business-To-Business Integration) を使用している場合、デフォルト Web アプリケーションのデプロイメント記述子に TransportServletFilter のエントリを追加する必要があります。ドメインを作成したときに（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」を参照）B2B 機能を含んだテンプレートを使用した場合、ドメインにはすでに TransportServletFilter のエントリがあります。

たとえば WLI ドメイン テンプレートでドメインを作成した場合、次のリストのように *DOMAIN_HOME* が作成したドメインのパス名を表す *DOMAIN_HOME* \applications\DefaultWebApp_myserver\WEB-INF\web.xml ファイルに TransportServletFilter が追加されます。

コード リスト 2-2 web.xml の TransportServerFilter のエントリ

```
<!-- WLI-B2Bi filter-begin.  DO NOT EDIT -->
<filter>
  <filter-name>TransportServletFilter</filter-name>
  <filter-class>com.bea.b2b.transport.http.TransportServletFilter</filter-class>
```

```
</filter>
<filter-mapping>
  <filter-name>TransportServletFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
<!-- WLI-B2Bi filter-end. -->
```

手順 8. セキュリティ レルム データの移行

独自のユーザまたはグループなどのエントリなどのデータを WebLogic Integration 2.1 または 2.1 SP1 セキュリティ レルムに追加した場合、2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した新しいドメインに同じデータを追加する必要があります。

警告： WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 ドメイン ディレクトリから新しい WebLogic Integration 7.0 ドメインに `fileRealm.properties` および `SerializedSystemIni.dat` ファイルをコピーしないでください。デフォルト システム ユーザ ID およびグループが変更されています。詳細については、2-11 ページの「WebLogic Integration のシステム ID」を参照してください。

RDBMS レルムから移行する

RDBMSRealm が使用されている WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 アプリケーションから移行する場合、FileRealm から RDBMSRealm へ（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成された）`config.xml` ファイルのドメインの指定されたレルムを変更する必要があります（コード リスト 2-3 に示す）`config.xml` ファイルの Realm 要素を、既存の WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 `config.xml` ファイルからの `RDBMSRealm`、`CachingRealm`、および `Realm` 要素に置き換える必要があります。

WebLogic Server のインスタンスを起動していない場合、生成された `config.xml` ファイルのコメント セクションには RDBMSRealm 要素が含まれます。`config.xml` ファイルに RDBMSRealm 要素を追加する場合、このセクショ

ンの区切り記号を削除できます。デフォルト データベースである PointBase は既にコンフィグレーションされています。既存の We b Logic Integtation 2.1 または WebLogic Integration 2.1 SP1 config.xml ファイルから、データベース コンフィグレーション情報をアップデートする必要があります。

コード リスト 2-3 置き換えられるレルム要素

```
<Realm CachingRealm="" FileRealm="myFileRealm" Name="myRealm"/>
```

アプリケーションのデータベース属性は、既に WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 config.xml ファイルでコンフィグレーションされています。コード リスト 2-4 は Oracle データベースのリストの例を示します。

コード リスト 2-4 追加される RDBMS 要素

```
<RDBMSRealm DatabaseDriver="oracle.jdbc.driver.OracleDriver"
DatabaseURL="jdbc:oracle:thin:@(description=(address=(host=
MY_ORACLE_SERVER)(protocol=tcp)(port=1521))(connect_data=
(sid=MY_ORACLE_SID)))" DatabaseUserName="scott"
DatabasePassword="tiger" Name="wlpiRDBMSRealm"
RealmClassName="com.bea.wlpi.rdbmsrealm.RDBMSRealm"
SchemaProperties="getGroupNewStatement=true;
removeUserFromGroup=DELETE FROM USERMEMBER WHERE USERID
= ? AND GROUPID = ?;getAcls=SELECT NAME, PRINCIPAL,
PERMISSION FROM ACLENTRIES ORDER BY NAME,
PRINCIPAL;addUserToGroup=INSERT INTO USERMEMBER
(USERID, GROUPID) VALUES ( ?, ? );getGroupMembersUsers
=SELECT USERMEMBER.USERID, PASSWORD FROM USERMEMBER,
WLSUSER WHERE GROUPID = ? AND USERMEMBER.USERID =
WLSUSER.USERID;newGroup=INSERT INTO WLSGROUP (GROUPID)
VALUES ( ? );addGroupToGroup=INSERT INTO GROUPMEMBER
(GROUPMEMBERID, GROUPID) VALUES ( ?, ? );newUser=INSERT
INTO WLSUSER (USERID, PASSWORD) VALUES ( ? , ?
);removeGroupFromGroup=DELETE FROM GROUPMEMBER WHERE
GROUPMEMBERID = ? AND GROUPID = ?;deleteGroup4=DELETE FROM
WLSGROUP WHERE GROUPID = ?;deleteUser3=DELETE FROM WLSUSER
WHERE USERID = ?;deleteGroup3=DELETE FROM USERMEMBER WHERE
GROUPID = ?;getPermissions=SELECT DISTINCT PERMISSION
FROM ACLENTRIES;deleteUser2=DELETE FROM USERMEMBER WHERE
USERID = ?;getPermission=SELECT DISTINCT PERMISSION FROM
ACLENTRIES WHERE PERMISSION = ?;getUser=SELECT USERID,
PASSWORD FROM WLSUSER WHERE USERID = ?;deleteGroup2=DELETE
FROM ACLENTRIES WHERE PRINCIPAL = ?;deleteGroup1=DELETE FROM
GROUPMEMBER WHERE GROUPID = ?;deleteUser1=DELETE FROM
ACLENTRIES WHERE PRINCIPAL = ?;getAclEntries=SELECT
```

```
NAME, PRINCIPAL, PERMISSION FROM ACLENTRIES WHERE NAME = ?
ORDER BY PRINCIPAL;getGroupMembersGroups=SELECT GROUPMEMBERID,
GROUPID FROM GROUPMEMBER WHERE GROUPID = ?;getGroups=
SELECT GROUPID FROM WLSGROUP ORDER BY GROUPID;
getGroup=SELECT GROUPID FROM WLSGROUP WHERE GROUPID
= ?;getUsers=SELECT USERID, PASSWORD FROM WLSUSER
ORDER BY USERID"/>

<CachingRealm BasicRealm="wlpiRDBMSRealm"
CacheCaseSensitive="true" Name="wlpiCachingRealm"/>

<Realm CachingRealm="wlpiCachingRealm" FileRealm=
"myFileRealm" Name="myRealm"/>
```

注意： コード リスト 2-4 の RDBMSRealm 要素は、config.xml ファイルで 1 つの連続する行である必要があります。コード リスト 2-4 では要素は読みやすさのために複数の行に分けられています。

RDBMSRealm からの移行は、Microsoft SQL Server および Oracle データベースのみでサポートされています。

注意： WebLogic Server 7.0 ではトラステッド リレーションシップの新しいセキュリティ モデルが使用されています。詳細については、3-8 ページの「トラステッド リレーションシップ」を参照してください。

手順 9. セキュリティのキーストアへの移行

WebLogic Integration 2.1 および WebLogic Integration 2.1 SP1 アプリケーションでは、関連するプライベート キーおよび証明書はファイル システムに格納されています。WebLogic Platform 7.0 はキーおよび証明書が格納されるキーストアを提供しています。WebLogic Integration アプリケーションが、SSL プロトコルで通信するようコンフィギュレーションされている、または SSL プロトコル メッセージの暗号化やデジタル署名を使用するようコンフィグレーションされているデリバリ チャネルのあるトレーディング パートナを含む場合、アプリケーションで新しいキーストアを使用することをお勧めします。

トレーディング パートナ間の通信のための SSL プロトコルの使用は、すべての WebLogic Integration B2B コラボレーション プロトコル（ebXML、XOCP、および RosettaNet）でサポートされています。メッセージの暗号化およびデジタル署名は RosettaNet 2.0 でのみサポートされています。

注意： WebLogic Integration 7.0 の場合、サポートされている キーストア プロバイダは Sun Microsystems の JKS（Java Keystore）のみです。

WebLogic Integration でキーストアを使用する方法の詳細については『*B2B Integration セキュリティの実装*』の「キーストアのコンフィグレーション」を参照してください。

すべての既存の証明書およびプライベート キーをファイル システムからキーストアにインポートして、新しいキーストアを使用するには、WebLogic Integration アプリケーションをコンバートします。コンバートするにはするには、次の手順を実行します。

1. WebLogic Integration ホーム ディレクトリに移動し、`setenv` スクリプトを実行して最上位レベルの WebLogic Integration 環境変数を設定します。これらのタスクを行うために実行するコマンドは、使用するプラットフォームによって異なります。
   - Windows:

     ```
     cd c:\bea\weblogic700\integration
     setEnv.cmd
     ```

   - UNIX:

     ```
     cd /home/joe/bea/weblogic700/integration
     . setenv.sh
     ```

2. 次のコマンドを行い、2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した新しい WebLogic Integration 7.0 ドメインに移動します。

   ```
   cd DOMAIN_HOME
   ```

   DOMAIN_HOME は、新しいドメインのパス名を表します。

3. テキスト エディタで、該当する起動スクリプト（Windows の場合 `startWebLogic.cmd` スクリプト、UNIX の場合 `startWebLogic` スクリプト）を開き、次の変更を加えます。

   a. 次のコードのように、WebLogic Server のインスタンスを開始する `java` コマンドを含む行を探します。

```
%JAVA_HOME%\bin\java ... weblogic.Server
```

b. Java システム プロパティ `wli.keystore.automigrate` を `java` コマンド ラインに追加し、次のリスト（太字）のように `true` と等しいプロパティを設定します。

```
%JAVA_HOME%\bin\java ... -Dwli.keystore.automigrate=true
weblogic.Server
```

c. Java システム プロパティ `wli.keystore.password` をプラットフォームに合わせて適切な `java` コマンド ラインに追加します。

Windows プラットフォームの場合、次のリストで太字表示されている文字列を入力します。

```
%JAVA_HOME%\bin\java ... -Dwli.keystore.automigrate=true
-Dwli.keystore.password=%KEYSTORE_PASS% weblogic.Server
```

UNIX プラットフォームの場合、次のリストで太字表示されている文字列を入力します。

```
$JAVA_HOME/bin/java ... -Dwli.keystore.automigrate=true
-Dwli.keystore.password=$KEYSTORE_PASS weblogic.Server
```

いずれのリストでも `KEYSTORE_PASS` は、キーストアのパスワードが含まれた環境変数を表します。コードリストに示すように、スクリプトにパスワードを含むのではなく、環境変数を使用してパスワードを指定することを強くお勧めします。パスワードはファイルにクリア テキストで保存しないでください。

`KEYSTORE_PASS` 環境変数で指定するパスワードは、手順 8 で指定したキーストアのパスワードと一致する必要があります。

d. Java システム プロパティ `wli.cakeystore.password` をプラットフォームに合わせて適切な `java` コマンド ラインに追加します。

Windows プラットフォームの場合、次のリストで太字表示されている文字列を入力します。

```
%JAVA_HOME%\bin\java ... -Dwli.keystore.automigrate=true
-Dwli.keystore.password=%KEYSTORE_PASS%
-Dwli.cakeystore.password=%CAKEYSTORE_PASS% weblogic.Server
```

UNIX プラットフォームの場合、次のリストで太字表示されている文字列を入力します。

```
$JAVA_HOME/bin/java ... -Dwli.keystore.automigrate=true
-Dwli.keystore.password=$KEYSTORE_PASS
-Dwli.cakeystore.password=$CAKEYSTORE_PASS weblogic.Server
```

いずれのリストでも CAKEYSTORE_PASS は、CA (Certificate Authority: 認証局) キーストアのパスワードが含まれた環境変数を表します。

CAKEYSTORE_PASS 環境変数で指定するパスワードは、手順 8 で指定したルート CA のパスワードと一致する必要があります。

e. WebLogic Integration アプリケーションで使用する各証明書に対して、使用しているプラットフォームの次のシステム プロパティを java コマンドラインに追加します。

Windows:

```
-D Key.certificateName.password=%KEY_PASS1%
```

UNIX:

```
-D Key.certificateName.password=$KEY_PASS1
```

いずれのプロパティ設定の場合でも、*certificateName* は証明書の名前を表し、KEY_PASS1 は証明書のプライベート キーのパスワードを含む環境変数を表します。これらのプロパティの設定は、WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 アプリケーションの証明書の設定と一致する必要があります。

アプリケーションで使用するすべてのセキュリティ証明書にこのシステム プロパティを追加します。たとえばそれぞれ（プライベート キー パスワードを定義するために）passcert1 および passcert2 という環境変数を持つ cert1 および cert2 という 2 つの証明書を指定した場合、次のリストのような startWebLogic.cmd スクリプト（Windows）の java コマンドラインが表示されます。

```
%JAVA_HOME%\bin\java ... -Dwli.keystore.automigrate=true
-Dwli.keystore.password=$KEYSTORE_PASS
-DKey.cert1.password=%passcert1%
-DKey.cert2.password=%passcert2% weblogic.Server
```

4. プライベート キー パスワードの環境変数、キーストアのパスワード、および手順 3 で定義したルート CA のパスワードを設定します。

KEYSTORE_PASS および CAKEYSTORE_PASS 環境変数の値は、手順 8 で指定したキーストアおよびルート CA キーストアののパスワードと一致している必要があります。

5. テキスト エディタで（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しいドメインの `config.xml` ファイルを開き、次の例に示すように `WebLogic Integration` アプリケーション要素の `Deployed` 属性の値として `false` を設定します。

   ```
   <Application Deployed="false" Name="WebLogic Integration"
   Path="c:/bea/weblogic700/integration/lib>" TwoPhase="true">
   ```

   この属性を `false` に設定した場合、`WebLogic Integration` アプリケーション（J2EE コンポーネントのコレクションが WebLogic Integration を構成する）は、新しいドメインの WebLogic Server のインスタンスが次回起動された場合にはデプロイされません。

6. 2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した新しい WebLogic Integration 7.0 ドメインで、プラットフォームに合わせて適切な手順を実行し、WebLogic Server のインスタンスを開始します。

- Windows システムの場合、2 種類の方法が存在します。
  - メニューからサーバを開始するには次の手順を実行します。

    [スタート | BEA WebLogic Platform 7.0 | User Projects | *`domain`* | Start Server] を選択します。
  - コマンド ラインで サーバを起動するには、次の手順を実行します。

    ```
    startWeblogic.cmd
    ```
- UNIX:

  ```
  startWebLogic
  ```

7. WebLogic Server Administration Console にログオンします。

   a. 新しいブラウザ ウィンドウを開きます。

   b. システムの WebLogic Server Administration Console の URL を入力します。実際に入力する URL は、システムによって異なります。入力フォーマットは次のとおりです。

      `http://`*`host`*`:`*`port`*`/console`

      WebLogic Server 管理 ログオン ページが表示されます。

c. WebLogic Server ユーザ名およびパスワードをクリックし、[Sign In] をクリックします。ここで入力したユーザ名およびパスワードは 2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」でドメインを作成したときに指定したユーザ名およびパスワードです。

BEA WebLogic Server ホームへようこそ、というページが表示されます。

注意： 詳細手順については『*WebLogic Integration の起動、停止およびカスタマイズ*』の「WebLogic Integration 管理ツールと設計ツール」にある「WebLogic Server Administration Console の起動」を参照してください。

8. キーストアを作成します。詳細については『*B2B Integration セキュリティの実装*』の「キーストアのコンフィグレーション」にある「作成するキーストア」を参照してください。

9. Sun Microsystems の JKS の WebLogic キーストア プロバイダのコンフィグレーションを行います。手順については『*B2B Integration セキュリティの実装*』の「キーストアのコンフィグレーション」にある「WebLogic キーストア プロバイダのコンフィグレーション」を参照してください。

10. プラットフォームに合わせて適切な手順を実行し、（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しい WebLogic Integration 7.0 ドメインの WebLogic Server インスタンスをシャットダウンします。

- Windows:

```
cd DOMAIN_HOME
stopWeblogic.cmd
```

- UNIX:

```
cd DOMAIN_HOME
stopWebLogic
```

いずれの `cd` コマンド ラインでも、*`DOMAIN_HOME`* は（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しいドメインのパス名を表し、*`domain`* は 2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成したディレクトリを表します。

11. テキスト エディタで（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しいドメインの `config.xml` ファイルを開き、次の例に示すように `WebLogic Integration` アプリケーション要素の `Deployed` 属性の値として `true` を設定します。

    ```
    <Application Deployed="true" Name="WebLogic Integration"
    Path="c:/bea/weblogic700/integration/lib>" TwoPhase="true">
    ```

    この属性を `true` に設定した場合、`WebLogic Integration` アプリケーション（J2EE コンポーネントのコレクションが WebLogic Integration を構成する）は、新しいドメインの WebLogic Server のインスタンスが次回起動された場合にデプロイされます。

12. プラットフォームに合わせて適切な手順を実行し、（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しい WebLogic Integration 7.0 ドメインの WebLogic Server インスタンスを開始します。

- Windows システムの場合、2 種類の方法が存在します。
  - メニューからサーバを開始するには次の手順を実行します。

    [スタート | BEA WebLogic Platform 7.0 | User Projects | *domain* | Start Server] を選択します。
  - コマンド ラインで サーバを起動するには、次の手順を実行します。

    ```
    startWeblogic.cmd
    ```
- UNIX:

    ```
    startWebLogic
    ```

    WebLogic Server のインスタンスが起動したときに、B2B エンジンはリポジトリから関連する証明書およびプライベート キーの場所を検索します。次にキーストアにそれらの証明書およびキーを移入します。

13. プラットフォームに合わせて適切な手順を実行し、（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しい WebLogic Integration 7.0 ドメインの WebLogic Server インスタンスをシャットダウンします。

- Windows:

    ```
    cd DOMAIN_HOME
    stopWeblogic.cmd
    ```
- UNIX:

```
cd DOMAIN_HOME
stopWebLogic
```

いずれの cd コマンド ラインの場合でも *DOMAIN_HOME* は 2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した新しいドメインのパス名を表します。

14. startWebLogic スクリプトから wli.keystore.automigrate システム プロパティを削除します。この手順は必須です。これにより次回 startWebLogic スクリプトが実行されたときに、B2B では再びファイル システムからキーストアへ証明書およびプライベート キーが移行されないようになります（この移行は 1 度のみ）。

    このプロパティを削除するには、次の手順を実行します。

    a. テキスト エディタでシステムの起動スクリプトを開きます。Windows の場合は startWebLogic.cmd で、UNIX の場合は startWebLogic です。

    b. スクリプトから wli.keystore.automigrate システム プロパティのみを削除します。

    警告： 手順 3 で追加した Key.*certificateName*.password、wli.cakeystore.password、および wli.keystore.password システム プロパティは削除しないでください。

    手順 3 で編集した同じスクリプトにこれらの変更を加えると次のリストのようにスクリプトは表示されます。

コード リスト 2-5 Windows の startWebLogic.cmd java コマンド例

```
%JAVA_HOME%\bin\java ... -Dwli.keystore.password=%KEYSTORE_PASS%
-Dwli.cakeystore.password=%CAKEYSTORE_PASS% -DKey.cert1.password=%passcert1%
-DKey.cert2.password=%passcert2% weblogic.Server
```

警告： 手順は、キーストアを最初に移入するために 1 回のみ実行します。

この手順にはサーバ証明書および関連するプライベート キーの移入への手順は含まれません。このサーバ証明書およびキーストアに関連するキーを追加する手順については『*B2B Integration セキュリティの実装*』の「キーストアのコンフィグレーション」の「キーストアの作成とサーバ証明書の追加」を参照してください。

手順 10. RosettaNet ワークフローの移行

WebLogic Integration アプリケーションに RosettaNet プロトコルを実装するワークフローが含まれている場合、WebLogic Integration 7.0 でアプリケーションを実行する前にこれらのワークフローに変更を加える必要があります。

注意： WebLogic Integration アプリケーションで ワークフローが使用されている場合は、3-11 ページの「RosettaNet スキーマの変更点」を参照してください。

WebLogic Integration 7.0 を RosettaNet ワークフローに対応させるには、次の手順を実行します。

1. WebLogic Integration ホーム ディレクトリに移動し、最上位 WebLogic Integration 環境変数を設定します。これらの変数を設定するには、プラットフォームに合わせて適切な `setenv` スクリプトを実行します。

   - Windows:

     ```
     cd c:\bea\weblogic700\integration
     setEnv.cmd
     ```

   - UNIX:

     ```
     cd /home/joe/bea/weblogic700/integration
     . setenv.sh
     ```

2. プラットフォームに合わせて適切な手順を実行し、（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しい WebLogic Integration 7.0 ドメインの WebLogic Server インスタンスを開始します。

   Windows システムの場合、2 種類の方法が存在します。

   - メニューからサーバを開始するには次の手順を実行します。

     [スタート | BEA WebLogic Platform 7.0 | User Projects | *`domain`* | *`servername`*] を選択します。

   - コマンド ラインで サーバを起動するには、次の手順を実行します。

     ```
     cd DOMAIN_HOME
     startWeblogic.cmd
     ```

   UNIX:

```
cd DOMAIN_HOME
startWebLogic
```

いずれの場合でも *DOMAIN_HOME* は「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した新しいドメインのパス名を表します。

3. 『*WebLogic Integration の起動、停止およびカスタマイズ*』の「WebLogic Integration 管理ツールと設計ツール」にある「Studio の起動」で説明するとおりに WebLogic Integration Studio を開始します。RosettaNet ワークフローのインスタンスが、現在 WebLogic Integration リポジトリに格納されている場合、手順 5 に移ります（2-11 ページの「手順 6. データベースの移行」を完了した後でも WebLogic Integration 2.1 リポジトリに格納されたテンプレートのインスタンスは使用可能）。
4. 次の手順を実行し、WebLogic Integration RosettaNet ワークフロー テンプレートを Studio にインポートします。
   a. [Studio] メニュー バーから [ツール (T) | パッケージをインポート ...] を選択します。
   b. [インポート : ファイルを選択] ウィンドウからアプリケーションでコンバートする RosettaNet ワークフロー テンプレートを含む JAR ファイルを選択します。[次へ] をクリックします。
   c. [対象オーガニゼーション] の下にあるドロップダウン リストを展開し、使用しているアプリケーションに該当するオーガニゼーションを選択します。
   d. [インポート] をクリックし、次に [閉じる] をクリックします。

   注意 : ワークフローのインポートの詳細は、『*WebLogic Integration Studio ユーザーズ ガイド*』の「ワークフロー パッケージのインポートとエクスポート」を参照してください。
5. 次の手順を実行し、WebLogic Integration RosettaNet ワークフロー テンプレートを Studio で開きます。
   a. 左ペインの [オーガニゼーション] の下にあるドロップダウン リストを展開し、使用しているアプリケーションに該当するオーガニゼーションを選択します。
   b. 左ペインで [テンプレート] フォルダを展開します。このアプリケーションのテンプレートがすべてリストされます。

c. コンバートするために左ペインの [テンプレート] フォルダを展開します。

d. テンプレート フォルダ内で、コンバートの対象となるフォルダを右クリックします（表示されている日付およびタイムスタンプによって識別する）。メニューが表示されます。

e. [開く] を選択します。

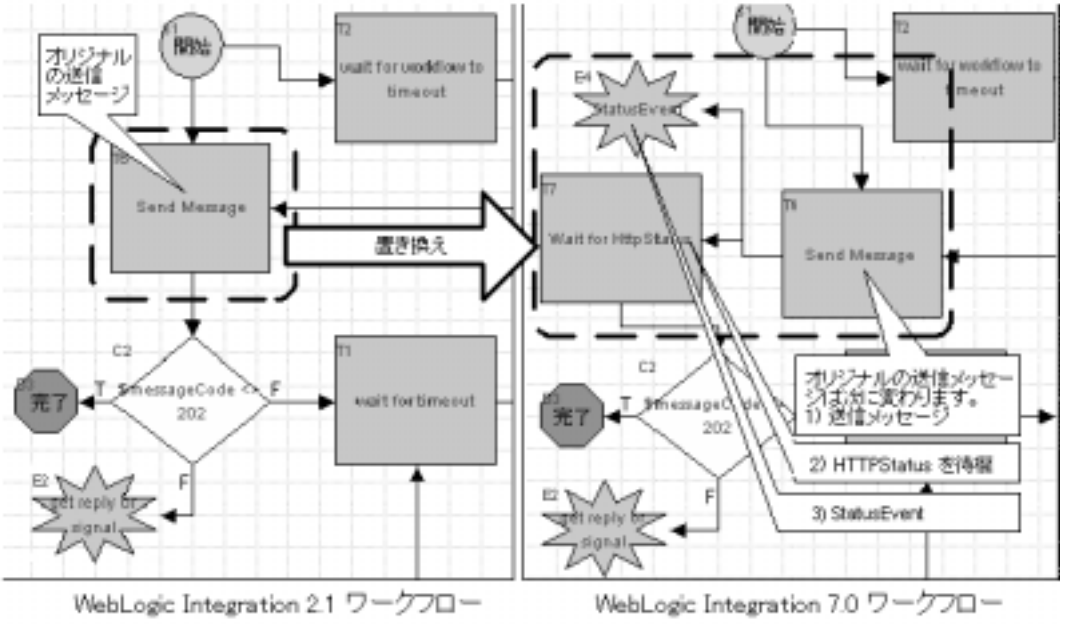
ワークフローを構成する開始ノード、タスク ノード、分岐ノード、およびイベント ノードが表示されます。

6. ワークフロー内の [ビジネス メッセージの送信] アクションのあるタスク ノードの各インスタンスを次の 3 つのノードに置き換えます。

- Send Business Message (タスク ノード)
- RosettaNet Status Event (イベント ノード)
- Wait for HTTP ステータス（タスク ノード）

これらのノードを追加する詳細手順は手順 6-a から 6-v で説明されています。次の図は、[ビジネス メッセージの送信] アクションでタスク ノードのすべてのインスタンスを代用した場合のワークフローの変更状態を示します。

図 2-3 WebLogic Integration 2.1 および WebLogic Integration 7.0 のワークフローの違い



WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 の RosettaNet ワークフローでは、Send Business Message タスクがメッセージを送信し、http ステータス コードを受信して、次のノードに進みます。WebLogic Integration 7.0 の RosettaNet ワークフローの場合、Send Business Message タスクがメッセージを送信し、イベント ノード（RosettaNet Status Event）が http ステータスを待ちます。ステータスを受信したときに、イベント ノードは、別のタスク ノード（Wait for HTTP Status）を終了と見なし、ワークフローは Wait for HTTP Status タスク ノードから進みます。

WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 の RosettaNet ワークフローの場合、Send Business Message タスクは同期となります。ワークフローの次のノードに進む前に http ステータスの応答を待ちます。WebLogic Integration 7.0 の RosettaNet ワークフローの場合、Send Business Message タスクは非同期です。ワークフローの次のノードに進む前に http ステータスの応答を待ちません。

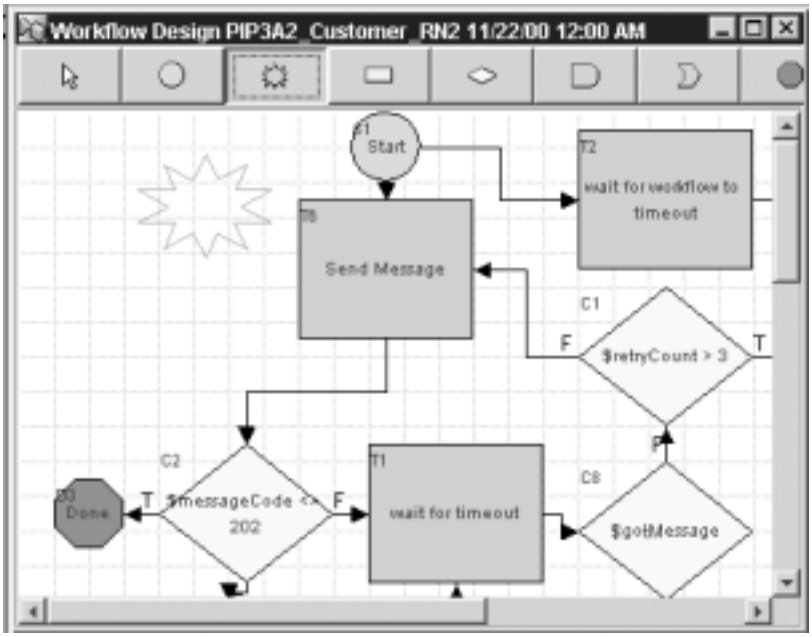
注意： 図 2-3 に示す PIP3A2_Customer ワークフローは、B2B RosettaNet Security サンプルから取得します。B2B RosettaNet Security サンプルは WebLogic Integration 7.0 で実行するためにコンバートされました。そのワークフローは次のファイルにあります。
SAMPLES_HOME/integration/samples/RN2Security/workflow/RN2Workflows.jar
このパス名では *SAMPLES_HOME* は、WebLogic Platform サンプル ディレクトリを表します。そこで使用可能なワークフローは、このセクションで説明する Send Message の変更例を提供します。WebLogic Integration の以前のリリースで提供される元の PIP3A2 ワークフローを変更していない場合、アプリケーションの PIP3A2 ワークフローを B2B RosettaNet Security サンプルで提供されている新しいワークフローに置き換えることができます。

WebLogic Integration 7.0 と対応の RosettaNet ワークフロー内の Send Business Message タスクのインスタンスを作成するには次の手順を実行します。

a. ワークフローから、コンバートする Send Business Action タスクを探します。

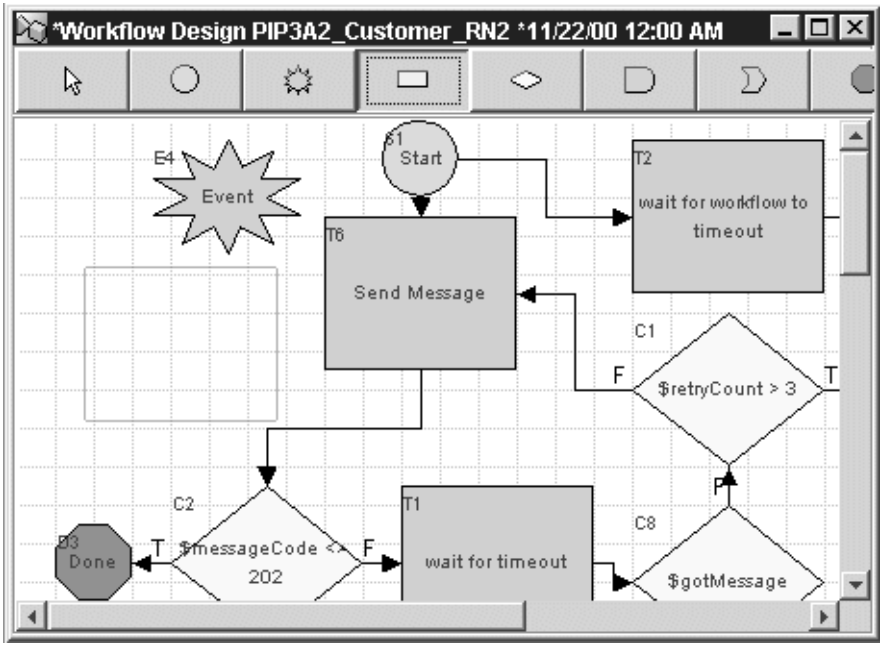
b. イベント ノードを作成します。ツール バーの [イベントを作成] をクリックし、次に Send Business Action タスクの付近にあるワークフロー図の場所をクリックします。

図 2-4 イベント ノードの作成



c. タスク ノードを作成します。ツール バーの [タスクを作成] をクリックし、次に Send Business Action タスクの付近にあるワークフロー図の場所をクリックします。

図 2-5 新しいタスク ノード

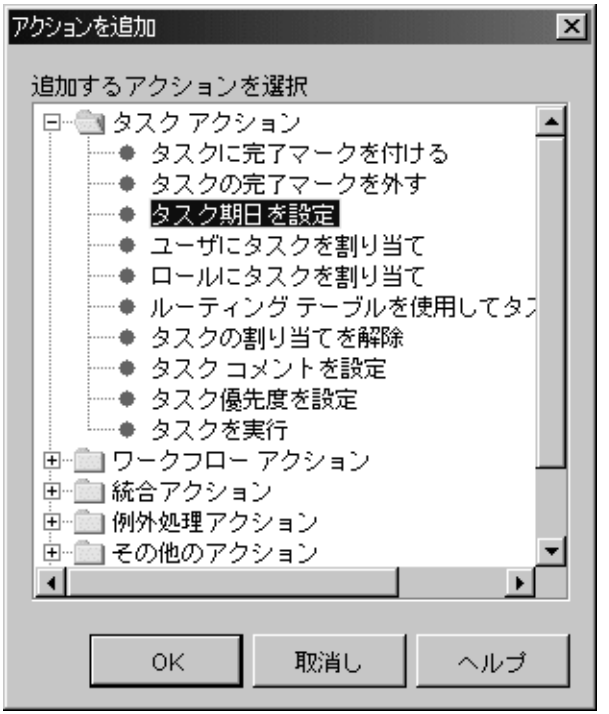


d. 作成したタスク ノードをダブルクリックします。[タスクのプロパティ] ウィンドウが表示されます。

図 2-6 [タスクのプロパティ] ウィンドウ

e. [Wait for Http Status] など、[タスク名] フィールドに説明的なタスク名を入力します。ウィンドウの [アクション] セクションにある [作成時] タブを選択して、[追加] をクリックします。[アクションを追加] ウィンドウが表示されます。

図 2-7 [アクションを追加] ウィンドウ



f. [タスク アクション] という名前のフォルダを展開し、[タスク期日を設定] を選択します。

[タスク期日を設定] ウィンドウが表示されます。

図 2-8 タイムアウト値の設定

g. Task という [期日を設定するタスク] 見出しの下に表示されているリストから、現在のタスクを選択します（手順 6-e の例で作業を行っている場合、タスク名は Wait for Http Status となる）。

[式に設定] フィールドに、応答を受信するタイムアウトの間隔として指定する時間を表す式を入力します。たとえばタイムアウトを 2 時間に設定するには図 2-8 に示す式を入力します。

`DateAdd()` メソッド `"h"` の 2 つ目の引数に注目してください。時間単位を指定します（時間）。3 つ目の引数は時間の量を指定します。図 2-8 に示す例では、指定されたタイムアウトの間隔は 2 時間です。

h. [期日に実行するアクション] タブを選択して [追加] をクリックします。

[アクションを追加] ウィンドウが表示されます。

i. [Task Actions] フォルダを展開し、[タスクに完了マークを付ける] を選択して [OK] をクリックします。

[タスクに完了マークを付ける] ウィンドウが表示されます。

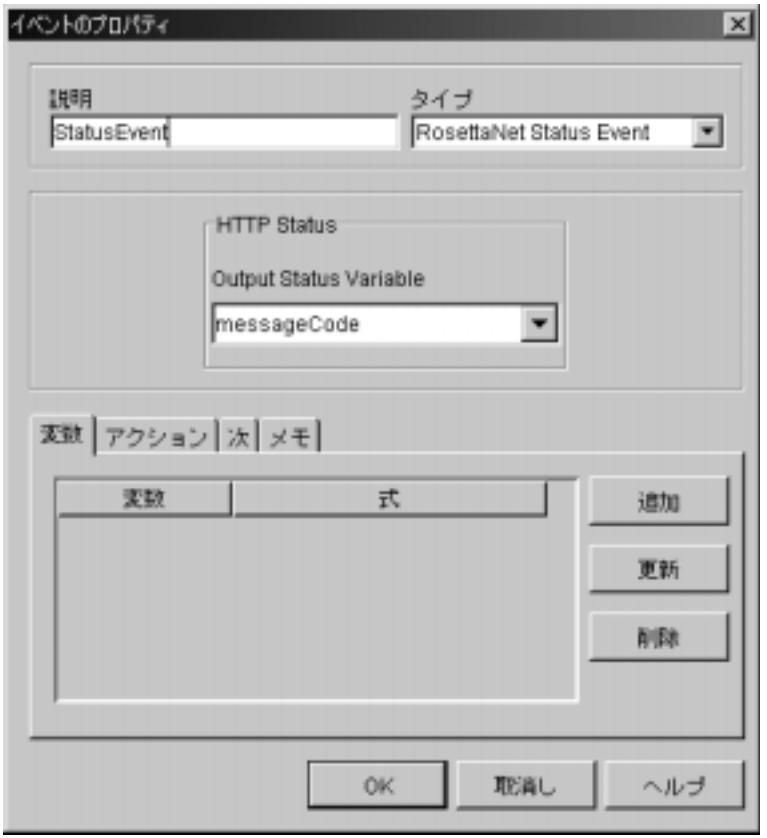
j. [完了マークを付けるタスク] リストから現在のタスクを選択します（手順 6-e にある例で作業を行っている場合、タスク名は Wait for Http Status となる）。[OK] をクリックします。

k. [タスクのプロパティ] ウィンドウで [OK] をクリックします。アプリケーションのワークフロー テンプレートの [ワークフロー設計] ウィンドウに戻ります。

l. [ワークフロー設計] ウィンドウから手順 6-b で作成したイベント ノードをダブルクリックします。[イベントのプロパティ] ウィンドウが表示されます。

m. [説明] フィールドに StatusEvent のような適切なイベント名を入力します。[タイプ] フィールドから [RosettaNet Status Event] を選択します。図 2-9 に示すように RosettaNet Status Event の特定のフィールドを反映するように [イベントのプロパティ] ウィンドウがアップデートされます。

図 2-9 RosettaNet Status イベント の [イベントのプロパティ] ウィンドウ



n. [Output Status Variable] フィールドから HTTP ステータスを格納する変数を選択します。

Send Business Message によってメッセージが送信されます。RosettaNet Status イベントでは、HTTP ステータスの応答を受信するまで待機し、次にその HTTP ステータスを HTTP ステータス出力ステータス変数に格納します。図 2-5 に示す PIP3A2_Customer_RN2 例では HTTP ステータス応答は StatusEvent によって `messageCode` 変数に格納されます。次に分岐ノードは変数 `messageCode` をテストし、HTTP が 202 と等しいかを判断します。202 の HTTP ステータスとは、処理のために要求が受け付けられたことを表しますが、処理は完了していません。

o. [イベントのプロパティ] ウィンドウから [アクション] タブを選択して [追加] をクリックします。[アクションを追加] ウィンドウが表示されます。

p. [タスク アクション] フォルダを展開し、[タスクに完了マークを付ける] を選択して [OK] をクリックします。[タスクに完了マークを付ける] ウィンドウが表示されます。

q. 手順 6-c で作成した タスク ノードを選択して [OK] をクリックします（手順 6-e にある例で作業を行っている場合、タスク名は Wait for Http Status となる）。

r. [イベントのプロパティ] ウィンドウで [OK] をクリックします。

s. ビジネス メッセージを分岐ノードに送信する既存のタスクからリンクを削除します（この例では、分岐ノードには `$messageCode<> 202` という名が付く）。

   ツール バーの [シェイプを選択] をクリックします。

   リンクを選択して〔Delete〕を押します。

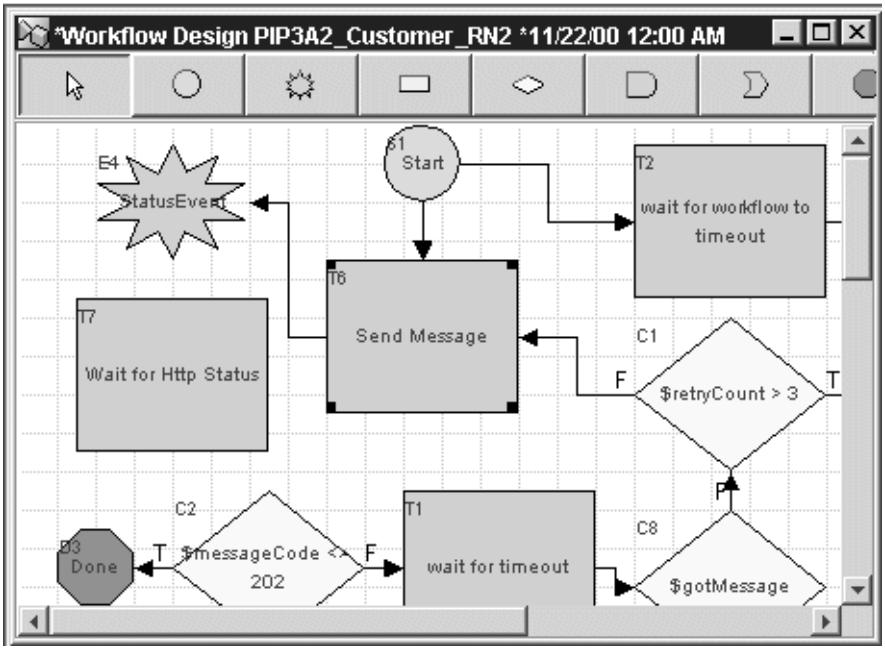
t. ビジネス メッセージを送信する既存のタスク（図 2-5 の Send Message タスク）を、手順 6-b で作成した新しいイベントにリンクします。

   ツール バーの [コネクタを描画] をクリックします。カーソルを既存の Send Message タスクの中央に移動します。

   [Send Message] タスクをクリックし、手順 6-b で作成したイベントまでドラッグして、マウス ボタンを放します（手順 6-e にある例で作業を行っている場合、イベント名は StatusEvent となる）。

   次の図に示すように、ワークフロー設計に（ビジネス メッセージを新しいイベントに送信するタスクから指す）矢印が追加されます。

図 2-10 ビジネス メッセージを状態イベントに送信するタスクのリンク



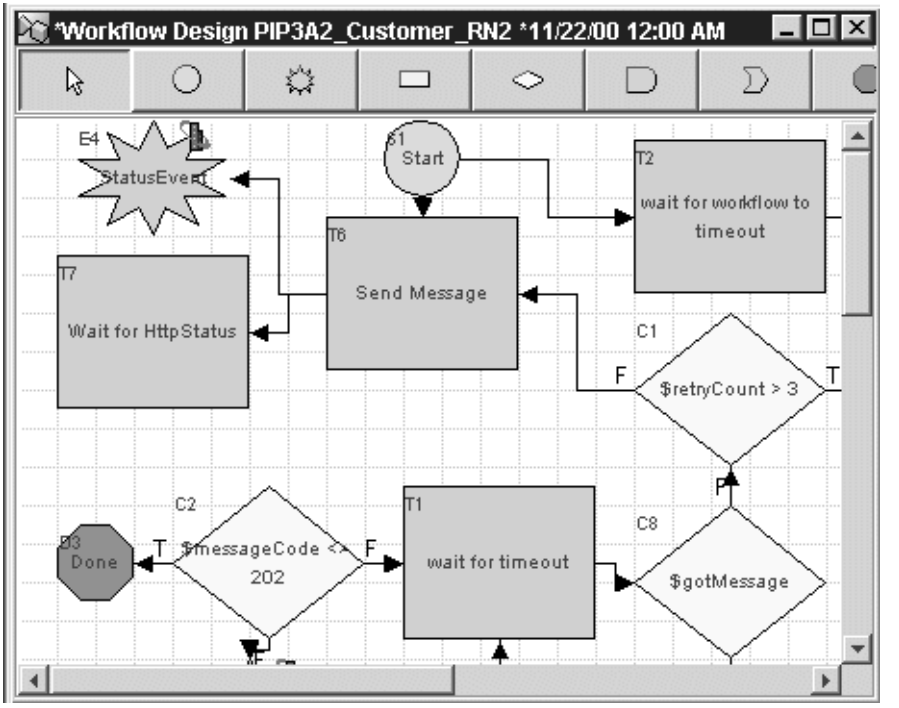
u. Send Message タスクを手順 6-c で作成したタスク ノードにリンクします。

ツール バーの [コネクタを描画] をクリックします。カーソルを既存の Send Message タスクの中央に移動します。

[Send Message] タスクをクリックし、手順 6-c で作成したタスクまでドラッグして、マウス ボタンを放します（手順 6-e にある例で作業を行っている場合、タスク名は Wait for Http Status となる）。

次の図に示すように、ワークフロー設計に（Send Message タスクから新しいタスクを指す）矢印が追加されます。

図 2-11 イベント ノードから新しいタスク ノードへのリンク



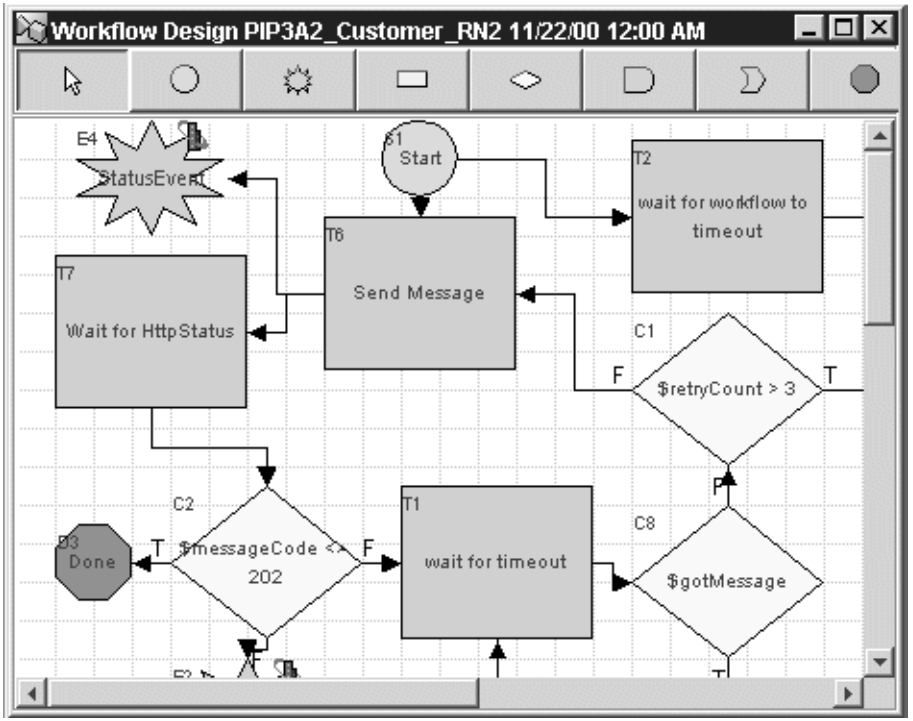
v. 新しいタスク ノードを、元の Send Message タスクに続くノードにリンクします（図 2-11 にある例では、新しい Wait for Http Status タスクを `$messageCode<> 202` という名の既存の分岐ノードにリンクする）。

ツール バーの ［コネクタを描画］をクリックします。カーソルを新しいタスク ノードの中央に移動します。

新しいタスク ノードをクリックし、分岐ノードまでドラッグしてマウス ボタンを放します（この例では、分岐ノードには `$messageCode<> 202` という名が付く）。

次の図に示すように、ワークフロー設計に新しいタスクから既存の分岐ノードを指す矢印が追加されます。

図 2-12 新しいタスクから既存の分岐ノードへのリンク



これで RosettaNet ワークフローの Send Business Message タスクのインスタンスのコンバートが完了し、WebLogic Integration 7.0 対応となりました。

7. RosettaNet アプリケーション ワークフローの Send Business Message タスクの各インスタンスに対して , 手順 6 a-v を繰り返します。
8. [ワークフロー設計] ウィンドウの右上の隅の [X] をクリックしてワークフローを保存します。[ワークフローが変更されています] ダイアログ ボックスが表示され、「直ちに変更を保存しますか ?」が表示されます。[はい] をクリックします。
9. アプリケーション JAR ファイルのすべてのワークフローに対して手順 5 から 8 を行った後に、JAR ファイルの新しいバージョンをエクスポートし、ワークフローをバックアップすることをお勧めします。手順は、次のとおりです。

a. [Studio] メニュー バーから [ツール (T) ｜ パッケージをエクスポート ...] を選択します。

b. [エクスポート : ファイルを選択] ウィンドウからコンバートされた RosettaNet ワークフロー テンプレートを含んだアプリケーション JAR ファイルを作成するための絶対パス名を入力します。[次へ] をクリックします。左ペインから、エクスポートするコンポーネントを選択します。[エクスポート] をクリックし、[閉じる] をクリックします。

注意 : ワークフローのエクスポートの詳細は、『*WebLogic Integration Studio ユーザーズ ガイド*』の「ワークフロー パッケージのインポートとエクスポート」を参照してください。

RosettaNet ワークフローのコンバートが完了しました。

手順 11. アプリケーション ビューのデプロイ

アプリケーションで WebLogic Integration のアプリケーション統合機能を使用している場合、次の自動デプロイ手順に従うまたは WebLogic Integration Application View Console を使用してアプリケーション ビューをデプロイする必要があります。アプリケーションでアプリケーション統合を使用していない場合には、この手順を省略します。

この節で説明する手順は、アプリケーションのアプリケーション ビューが 2-11 ページの「手順 6. データベースの移行」で WebLogic Integration 7.0 にコンバートされたデータベースに格納されていることを前提としています。新しいアプリケーション ビューは作成されません。新しい環境に対して正しいアプリケーション ビュー設定が行われていることを確認して、既存のアプリケーション ビューをデプロイします。

警告 : アプリケーション ビューをデプロイする前に、(2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した) 新しいドメインのアプリケーション統合アダプタ EAR ファイルのコンフィグレーションを行う必要があります。手順については、3-3 ページの「Application Integration アダプタ EAR ファイルのコンフィグレーション」を参照してください。

アプリケーション ビューをデプロイするには、次の手順のいずれかを実行します。

- Application View Console を使用してデプロイを行う ― イベント ルータ URL をアップデートする、またはアプリケーション ビューをクラスタ環境にデプロイする場合、この手順を実行します。
- 自動デプロイ アプリケーション ビュー

Application View Console を使用してデプロイを行う

Application View Console を使用してアプリケーション ビューをデプロイするには、次の手順を実行します。

1. WebLogic Integration ホーム ディレクトリに移動し、`setenv` スクリプトを実行して最上位レベルの WebLogic Integration 環境変数を設定します。これらのタスクを行うために実行するコマンドは、使用するプラットフォームによって異なります。
   - Windows:

     ```
     cd c:\bea\weblogic700\integration
     setEnv.cmd
     ```

   - UNIX:

     ```
     cd /home/joe/bea/weblogic700/integration
     . setenv.sh
     ```

2. プラットフォームに合わせて適切な手順を実行し、(2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しい WebLogic Integration 7.0 ドメインの WebLogic Server インスタンスを開始します。

- Windows システムの場合、2 種類の方法が存在します。
  - メニューからサーバを開始するには次の手順を実行します。

    [スタート | BEA WebLogic Platform 7.0 | User Projects | *domain* | Start Server] を選択します。

  - コマンド ラインで サーバを起動するには、次の手順を実行します。

```
cd DOMAIN_HOME
startWeblogic.cmd
```

- UNIX:

```
cd DOMAIN_HOME
startWebLogic
```

いずれの cd コマンド ラインの場合でも *DOMAIN_HOME* は「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した新しいドメインのパス名を表します。

3. Application View Console にログオンします。

   a. 新しいブラウザ ウィンドウを開きます。

   b. 該当するシステムの Application View Console の URL を入力します。実際に入力する URL は、システムによって異なります。入力フォーマットは次のとおりです。

      `http://host:port/wlai`

      [Application View Console Logon] 画面が表示されます。

   c. WebLogic Server ユーザ名およびパスワードをクリックし、[Login] をクリックします。

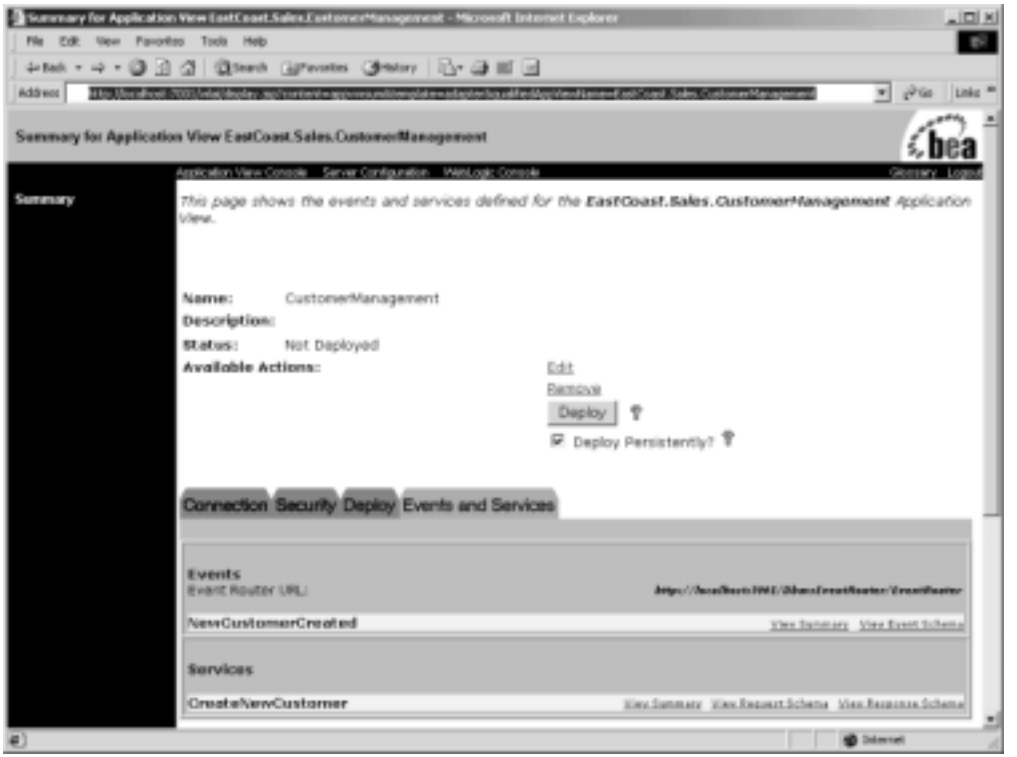
      Application View Console が表示されます。Console にアプリケーション ビューのリストが表示されます。

   注意： 詳細手順については、『*Application Integration ユーザーズ ガイド*』の「Application View の定義」にある「手順 1: Application View Console へのログオン」を参照してください。

4. リスト内のアプリケーション ビューをダブルクリックします。アプリケーションのエンド ノードに到達するまでクリックします。アプリケーションが現在デプロイされている場合、[Undeploy] をクリックします。

   選択されたアプリケーション ビューの [Summary] 画面が表示されます。次の図は、3-3 ページの「Application Integration アダプタ EAR ファイルのコンフィグレーション」で説明した例に基づいた EastCoast.Sales.CustomerManagement アプリケーション ビューのサマリ ページを示します。

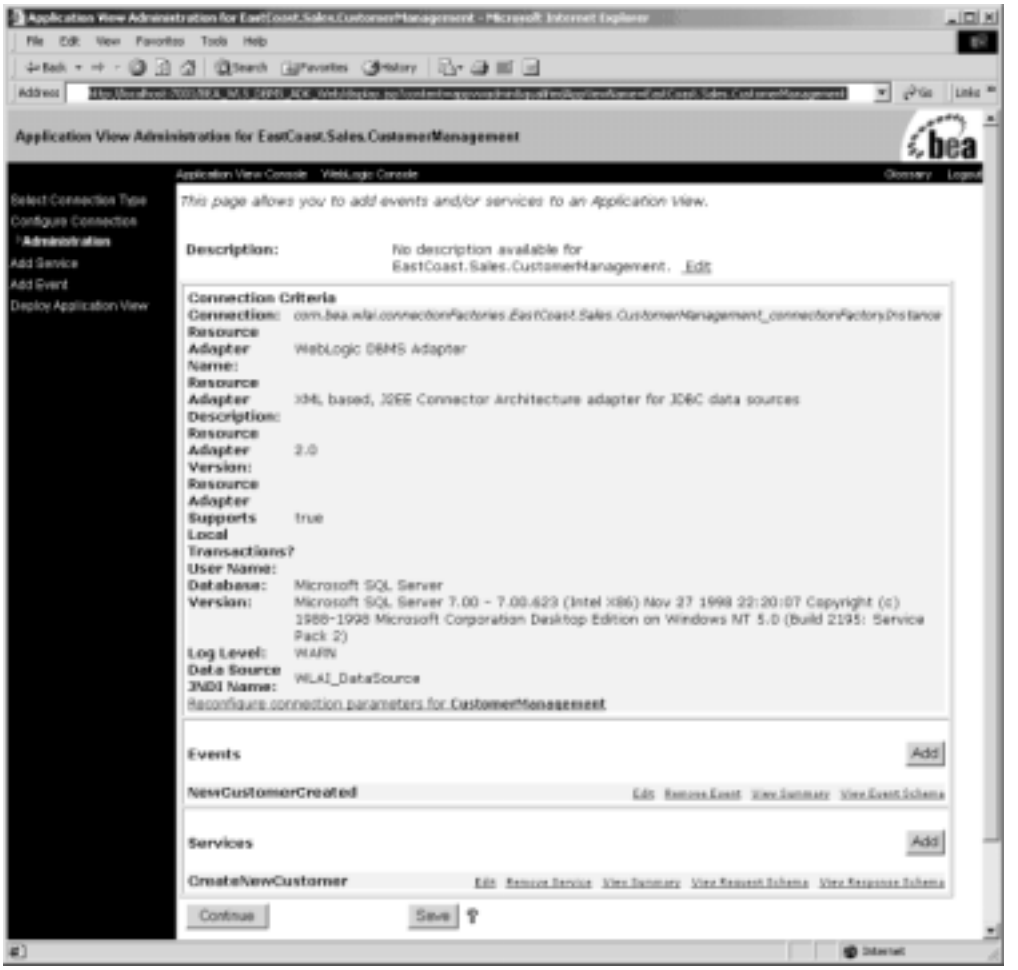
図 2-13 EastCoast.Sales.CustomerManagement アプリケーション ビュー



5. [Edit] をクリックします。

   [Application View Administration] 画面が表示されます。

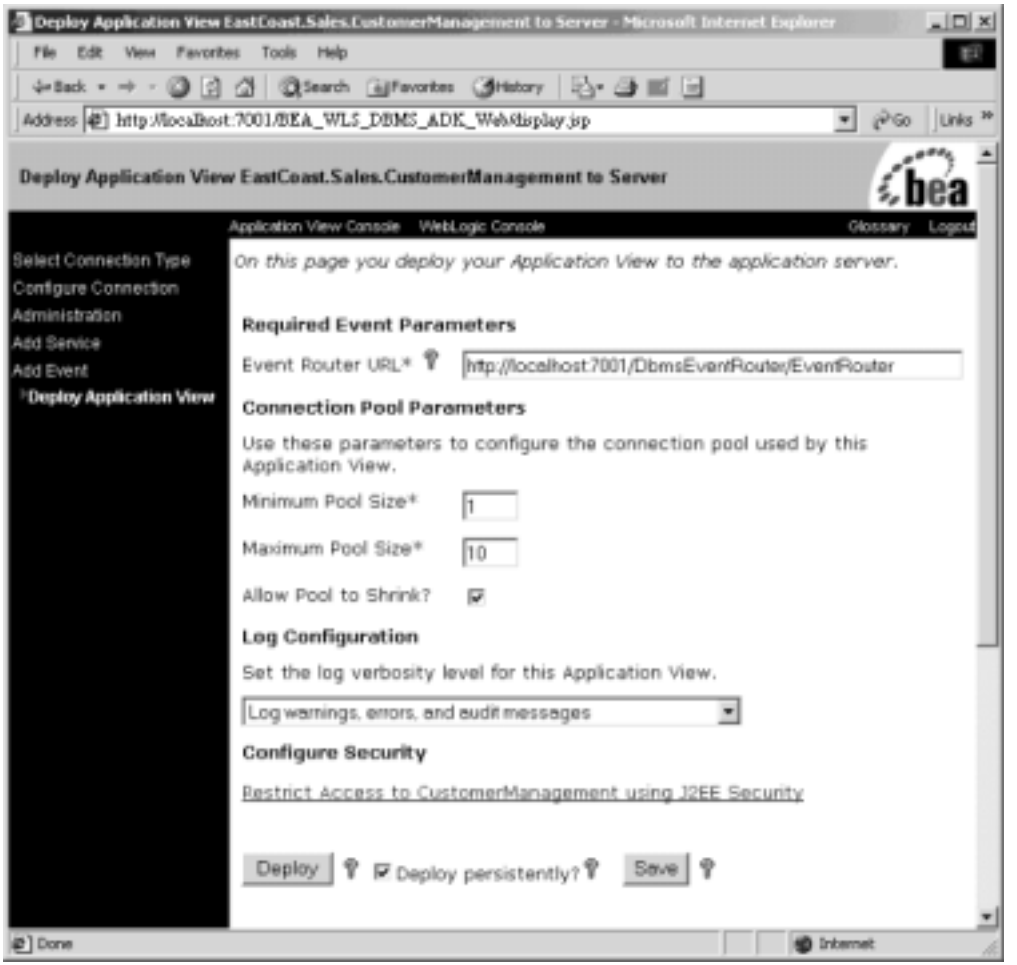
図 2-14 [Application View Administration] 画面



6. 右ペインの [Continue] をクリックします。

   [Deploy Application View] 画面が表示されます。

図 2-15 [Deploy Application View]



7. 使用している WebLogic Integration 7.0 環境に対し、イベント ルータ URL が正しいことを確認します。[Event Router URL] フィールドに必要な変更を加えます。
8. [Deploy] をクリックします。

自動デプロイ アプリケーション ビュー

この手順では、2-11 ページの「手順 6. データベースの移行」で WebLogic Integration 7.0 にコンバートされたデータベースに格納されたアプリケーション ビューを自動的にデプロイします。

アプリケーション ビューをデプロイする手順は次のとおりです。

1. （「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しいドメインの WebLogic Integration 7.0 `wlai.properties` ファイルを、次の手順を実行してバックアップします。

   a. *`DOMAIN_HOME`*`\wlai` ディレクトリに移動します（*`DOMAIN_HOME`* は、「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成したドメインのパス名を表す）。

   b. プラットフォームに合わせて適切な手順を実行し `wlai.properties` ファイルの名前を `wlai.properties.70` に変更します。

   - Windows:

     ```
     rename wlai.properties wlai.properties.70
     ```

   - UNIX:

     ```
     mv wlai.properties wlai.properties.70
     ```

2. WebLogic Integration 2.1 `wlai.properties` ファイルを（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しいドメインにコピーします。たとえば、アプリケーションで WebLogic Integration 2.1 EAI Domain が使用された場合、`wlai.properties` ファイルを *`WLI_HOME`*`\config\eaidomain\wlai` ディレクトリから *`DOMAIN_HOME`*`\wlai` ディレクトリにコピーします。*`WLI_HOME`* は WebLogic Integration 2.1 ホーム ディレクトリを表し、*`DOMAIN_HOME`* は「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成したドメインのパス名を表します。

3. テキスト エディタで `wlai.properties` ファイルを開きます。`wlai.appView.deploy` プロパティの値として、デプロイするアプリケーション ビューのコンマで区切られたリストを指定します。たとえば、`EastCoast.Sales.CustomerManagement` および

WestCoast.Sales.CustomerManagement アプリケーション ビューをデプロイするには、次のコード リストに示す wlai.appView.deploy プロパティを設定します。

```
wlai.appView.deploy=EastCoast.Sales.CustomerManagement,
WestCoast.Sales.CustomerManagement
```

4. wlai.properties ファイルから wlai.appView.deploy プロパティ以外のすべての他のプロパティを削除します。
5. プラットフォームに合わせて適切な手順を実行し、2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した新しい WebLogic Integration 7.0 ドメインの WebLogic Server インスタンスを開始します。

- Windows システムの場合、2 種類の方法が存在します。
  - メニューからサーバを開始するには次の手順を実行します。

    [スタート | BEA WebLogic Platform 7.0 | User Projects | *domain* | Start Server] を選択します。
  - コマンド ラインで サーバを起動するには、次の手順を実行します。

    ```
    startWeblogic.cmd
    ```

  WebLogic Server のインスタンスが起動されると、リストされたアプリケーション ビューは自動的にデプロイされます。

6. プラットフォームに合わせて適切な手順を実行し、（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しい WebLogic Integration 7.0 ドメインの WebLogic Server インスタンスをシャットダウンします。

- Windows:

  ```
  cd DOMAIN_HOME
  stopWeblogic.cmd
  ```

- UNIX:

  ```
  cd DOMAIN_HOME
  stopWebLogic
  ```

  いずれの cd コマンド ラインでも、*DOMAIN_HOME* は（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しいドメインのパス名を表し、*domain* は 2-6 ページの「手順

5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成したディレクトリを表します。

7. 新しいドメインの WebLogic Integration 7.0 wlai.properties ファイルを、次に手順を実行して復元します。

   a. *DOMAIN_HOME*\wlai ディレクトリに移動します（*DOMAIN_HOME* は、「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成したドメインのパス名を表す)。

   b. プラットフォームに合わせて適切な手順を実行し wlai.properties ファイルの名前を wlai.properties.21 に変更します。

   - Windows:

     ```
     rename wlai.properties wlai.properties.21
     ```

   - UNIX:

     ```
     mv wlai.properties wlai.properties.21
     ```

   c. プラットフォームに合わせて適切な手順を実行し wlai.properties.70 ファイルを wlai.properties にコピーします。

   - Windows:

     ```
     rename wlai.properties.70 wlai.properties
     ```

   - UNIX:

     ```
     mv wlai.properties.70 wlai.properties
     ```

手順 12. WebLogic Integration アプリケーションの開始およびテスト

WebLogic Integration 7.0 を使用してアプリケーションを開始、実行およびテストします。

WebLogic Integration 7.0 サーバを開始したときに次のような例外が発生した場合、SerializedSystemIni.dat ファイルを再生成する必要があります。

```
weblogic.security.internal.FileUtilsException: Couldn't rename
DOMAIN_HOME\SerializedSystemIni.dat to
DOMAIN_HOME\SerializedSystemIni.dat42698.old
```

このメッセージでは *DOMAIN_HOME* は 2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した新しいドメインのパス名を表します。

新しい SerializedSystemIni.dat ファイルの生成方法については 3-8 ページの「新しい SerializedSystemIni.dat ファイルの生成」を参照してください。

WebLogic Integration 2.1 リポジトリ データ ファイルをインポートする

2-11 ページの「手順 6. データベースの移行」では、データベースに格納されている WebLogic Integration 2.1 リポジトリ データは WebLogic Integration 7.0 フォーマットにコンバートされました。この節は、WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 のリポジトリ ファイルに格納された追加のリポジトリ情報をインポートする場合に必要となります。

2-11 ページの「手順 6. データベースの移行」でのデータベースの移行の一部として、wlisystem という新しいシステム ID が wlisystem のパスワードとともに作成されました。新しい ID は WebLogic Integration リポジトリおよびセキュリティ レルムの両方に格納されました。wlisystem システム ID は、WebLogic Integration 2.1 および WebLogic Integration 2.1 SP1 の wlcsystem および wlpisystem システム ID を置き換えます。

WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 リポジトリ データ ファイルを WebLogic Integration 7.0 リポジトリにインポートする前に リポジトリ データ ファイルの LC 要素のシステム パスワード属性を wlisystem の現在のパスワードに反映させるために変更する必要があります（WebLogic Integration 2.1 および WebLogic Integration 2.1 SP1 では、LC 要素のシステム パスワード属性によって指定された値は、wlcsystem システム ID のパスワード）。

次のリストに示すように、wlisystem システム ID のパスワードを反映するよう、リポジトリ データ ファイルの WLC 要素のシステム パスワード属性を設定します（wlisystem システム ID のパスワードは、B2B Console を使用して wlisystem パスワードを変更した場合を除き、wlisystem ）。

コード リスト 2-6 システム パスワードの設定

```
<?xml version="1.0"?>
<!DOCTYPE wlc SYSTEM "WLC.dtd">
<wlc
       large-msg-support-on="OFF"
       large-msg-min-size="0"
       large-msg-location="c:\temp"
       system-password="wlisystem"
       >
```

WebLogic Integration リポジトリ およびセキュリティ レルムの両方で wlisystem に対して同じパスワードを指定する必要があります。これらの 2 つのコピーを同期しないと、WebLogic Integration の WebLogic Integration インスタンスを起動したときに問題が発生する場合があります。これらの 2 つのコピーが同期されていることを確認するために、パスワードを変更するたびに B2B Console を使用してください。変更を加えると、B2B Console はセキュリティ レルムおよび WebLogic Integration リポジトリの両方でパスワードが保存されます。

リポジトリ データ ファイルをインポートすると、システム パスワード属性の値が、リポジトリの wlisystem システム ID のパスワードをオーバーライドします。セキュリティ レルムの wlisystem システム ID のパスワードはアップデートされません。WebLogic Integration Administration Console を使用して wlisystem パスワードを変更した場合、セキュリティ レルムでのみ wlisystem のパスワードが変更されます。

3　移行に関するその他のトピック

この章の内容は以下のとおりです。

- クラスタ化
- Application Integration アダプタ EAR ファイルのコンフィグレーション
- トラステッド リレーションシップ
- RosettaNet スキーマの変更点

クラスタ化

クラスタ化は、BPM (Business Process Management) によってサポートされており、アプリケーション統合機能は、WebLogic Integration 2.1、2.1 SP1、および 7.0 で提供されています。7.0 クラスタ アプリケーションのコンフィグレーション要件は、2.1 または 2.1 SP1 クラスタ アプリケーションの要件とは異なっており、その結果 2.1 および 7.0 リリースの `config.xml` ファイルのクラスタ化セクションが異なります。また、WebLogic Integration 7.0 は WebLogic Integration によって提供された B2B 機能のクラスタ化をサポートしています。

WebLogic Integration クラスタ アプリケーションをリリース 2.1 または 2.1 SP1 からリリース 7.0 に移行する場合、まずクラスタ化が使用可能な状態で新しい WebLogic Integration ドメインを作成し、次に 2-14 ページの「手順 7. WebLogic Integration アプリケーションのコンポーネントの移行」に記載する（JMS キューなどの）アプリケーション固有のエンティティを移行する必要があります。クラスタ化を使用可能にした状態で WebLogic Integration ドメインを作成する方法についての詳細は『*WebLogic Integration ソリューションのデプロイメント*』の「クラスタ デプロイメントのコンフィグレーション」を参照してください。

クラスタ化に関連するコンフィグレーション要件は、次の BPM リソースではリリース 7.0 で変更されました。

- JMS 接続ファクトリ

- JMS サーバ
- WebLogic Integration 7.0 の分散送り先である JMS 送り先 BPM クラスタ コンフィグレーションでは次の分散送り先が必要です。
  - BPM 監査トピック（`com.bea.wli.bpm.AuditTopic`)
  - BPM エラー トピック（`com.bea.wli.bpm.ErrorTopic`)
  - BPM 通知トピック（`com.bea.wli.bpm.NotifyTopic`)
  - BPM イベント キュー（`com.bea.wli.bpm.EventQueue`）
  - BPM イベント検証キュー（`com.bea.wli.bpm.ValidatingEventQueue`）
  - wli エラー キュー（`com.bea.wli.FailedEventQueue`)

クラスタ化に関連するコンフィグレーション要件は、次のアプリケーション統合リソースではリリース 7.0 で変更されました。

- JMS 接続ファクトリ
- JMS サーバ
- WebLogic Integration 7.0 の分散送り先である JMS 送り先アプリケーション統合クラスタ コンフィグレーションでは次の分散送り先が必要です。
  - wlai イベント キュー（`com.bea.wlai.EVENT_QUEUE`）
  - wlai イベント トピック（`com.bea.wlai.EVENT_TOPIC`）
  - wlai 非同期要求キュー（`com.bea.wlai.ASYNC_REQUEST_QUEUE`）
  - wlai 非同期応答キュー（`com.bea.wlai.ASYNC_RESPONSE_QUEUE`）
  - wli エラー キュー（`com.bea.wli.FailedEventQueue`)

詳細については『*WebLogic Integration ソリューションのデプロイメント*』の「WebLogic Integration クラスタについて」にある「JMS リソース」を参照してください。

Application Integration アダプタ EAR ファイルのコンフィグレーション

この節では、サンプル WebLogic Integration 7.0 アプリケーションで使用する Application Integration アダプタの EAR ファイルのコンフィグレーション手順を説明します。このサンプルは次にシナリオに基づきます。

Acme という WebLogic Integration 2.1 の顧客が WebLogic Integration アプリケーションを WebLogic Integration 7.0 に移行するとします。Acme アプリケーションには、ACME_DBMS というリレーショナル データベース アダプタを使用する EastCoast.Sales.CustomerManagement というアプリケーション ビューが 1 つ含まれています。ACME_DBMS アダプタは Acme の IT 開発者によって開発されました。

Acme アプリケーションの Application Integration アダプタ EAR ファイルのコンフィグレーションを行うには、次いずれかを実行します。

- 新しいドメインのコンフィグレーション ファイルを編集する
- WebLogic Server Deployer を呼び出す
- クラスタ環境でアダプタ EAR ファイルをデプロイする場合、『*WebLogic Integration ソリューションのデプロイメント*』の「WebLogic Integration クラスタについて」にある「アダプタのデプロイメント」の手順に従います。

新しいドメインのコンフィグレーション ファイルを編集する

（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しいドメインの `config.xml` ファイルを開き、アダプタのコンフィグレーション情報を追加します。手順は、次のとおりです。

1. テキスト エディタでアダプタ EAR ファイルをデプロイした WebLogic Integration 2.1 アプリケーション `config.xml` ファイルを開きます。

2. 次の Acme の例のリストに示すとおり、（Application 要素の一部として）アダプタ EAR ファイルをデプロイするファイルのセクションをコピーします。

コード リスト 3-1 アダプタ EAR ファイルの WebLogic Integration 2.1 のデプロイ

```
<Application Deployed="true" Name="ACME_DBMS" Path="d:\ACME_DBMS.ear">
    <ConnectorComponent Name="ACME_DBMS"
        Targets="myserver"
        URI="ACME_DBMS.rar"/>
    <WebAppComponent Name="DbmsEventRouter"
        Targets="myserver"
        URI="DbmsEventRouter.war"/>
    <WebAppComponent Name="ACME_DBMS_Web"
        Targets="myserver"
        URI="ACME_DBMS_Web.war"/>
</Application>
```

3. （2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しい WebLogic Integration 7.0 ドメインの `config.xml` ファイルを開き、Application 要素を貼り付けます。

4. 次の図に太字で表示されているとおり、Application 要素に `TwoPhase` 属性を追加します。この要素の値を `"true"` に設定します。

コード リスト 3-2 Application 要素への TwoPhase 属性の追加

```
<Application Deployed="true" Name="ACME_DBMS" TwoPhase="true"
Path="d:\ACME_DBMS.ear">
```

5. EAR ファイルの場所が変更された場合、次の図の太字で表示されたとおり、アダプタの `Application` 要素の `Path` 属性を更新します。

コード リスト 3-3 EAR ファイルのパスの更新

```
<Application Deployed="true" Name="ACME_DBMS" TwoPhase="true"
Path="c:\adk\ACME_DBMS.ear">
```

警告： アダプタ EAR ファイルが WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 インストールのホーム ディレクトリにある場合（たとえば、Windows システムの %WLI_HOME%\lib）、c:\adk などの WebLogic Integration 2.1 インストール外の場所に移動することをお勧めします。

6. アダプタの `Application` 要素の `Targets` 要素の設定を（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した）新しいドメインのサーバ名と一致するよう更新します。たとえば、ドメインが WLI ドメイン テンプレートから作成された場合、サーバ名は次のリストの例で太字表示されているとおり、`myserver` に設定されます。

コード リスト 3-4 Config.xml ファイルからのサーバ名の取得

```
<Domain Name="mydomain">
   <!-- Domain log -->
   <Log FileName="c:/bea/user_projects/mydomain/logs/mydomain.log"
Name="mydomain"/>
   <Server ListenPort="7001"
      Name="myserver"
      TransactionLogFilePrefix="c:/bea/user_projects/mydomain/logs/"
      NativeIOEnabled="true">
.
.
.
</Domain>
```

次のリストに太字で表示されているように、アダプタの `Application` 要素の `Targets` 属性の設定を更新します。

コード リスト 3-5 Targets 属性の更新

```
<Application Deployed="true" TwoPhase="true" Name="ACME_DBMS"
Path="c:\adk\ACME_DBMS.ear">
    <ConnectorComponent Name="ACME_DBMS"
        Targets="myserver"
        URI="ACME_DBMS.rar"/>
    <WebAppComponent Name="DbmsEventRouter"
        Targets="myserver"
        URI="DbmsEventRouter.war"/>
    <WebAppComponent Name="ACME_DBMS_Web"
        Targets="myserver"
        URI="ACME_DBMS_Web.war"/>
</Application>
```

WebLogic Server Deployer を呼び出す

WebLogic Server Deployer コマンド ライン ユーティリティを使用してアダプタ EAR をデプロイします。たとえば、Windows システムの ACME_DBMS.ear ファイルをデプロイするには、次のコマンドを使用します。

```
java -classpath %BEA_HOME%\weblogic700\lib\weblogic.jar
weblogic.Deployer -adminurl
http://wls_admin_host:wls_admin_port -user wls_admin_user
-password wls_admin_pass -verbose -stage -activate -source
c:\adk\ACME_DBMS.ear -name ACME_DBMS -targets server_name
```

表 3-1 Deployer の例で使用される代替可能な文字列

文字列	説明
BEA_HOME	Windows システムの c:\bea などの、BEA 製品ソフトウェアがインストールされるディレクトリ。
wls_admin_host	使用されている WebLogic Server インスタンスのホスト名または IP アドレス。
wls_admin_port	使用される WebLogic Server インスタンスのポート番号。

文字列	説明
wls_admin_user	WebLogic Server Adminstration Console にログインするために使用するユーザ名（この名前は 2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」でドメインを作成するためにコンフィグレーション ウィザードを使用したときに指定したユーザ名）。
wls_admin_pass	WebLogic Server Adminstration Console にログインするためのパスワード（2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」でドメインを作成するためにコンフィグレーション ウィザードを使用したときに指定したパスワード）。
server_name	2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成したドメインのサーバ名。

Deployer ユーティリティの詳細については、『*WebLogic Server アプリケーションの開発*』の「WebLogic Server デプロイメント」にある「デプロイメント ツールおよび手順」を参照してください。このマニュアルは、BEA WebLogic Server マニュアル セットの次の URL にあります。

```
http://edocs.beasys.co.jp/e-docs/wls/docs70/programming/deploying.html
```

警告： DBMS Sample Adapter は WebLogic Integration のリリース 7.0 で変更されました。これにより、このアダプタのアプリケーション ビューに影響する場合があります。詳細については『*アダプタの開発*』の「WebLogic Integration 7.0 へのアダプタの移行」にある「要求データを必要としないサービスに対する DBMS サンプル アダプタの変更」を参照してください。

Application Integration アダプタの移行の詳細については、『*アダプタの開発*』の「WebLogic Integration 7.0 へのアダプタの移行」を参照してください。

トラステッド リレーションシップ

WebLogic Server トラステッド リレーションシップのセキュリティ モデルは、WebLogic Server のリリース 7.0 で変更されました。（クライアントとして機能している）WebLogic 7.0 サーバは、別のドメインの別の WebLogic 7.0 サーバを認証します。いずれのドメインも `SecurityConfigurationMBean` に同じ Credential 属性を持つ必要があります。異なる場合、次の例外が送出されます。`java.lang.SecurityException Invalid Subject.`

次のいずれかの方法でドメインの Credential 属性を変更できます。

- WebLogic Server Administration Console を使用して資格を変更します。詳細については、『*BEA WebLogic Server 管理者ガイド*』を参照してください。このマニュアルは、BEA WebLogic Server マニュアル セットの次の URL にあります。

  `http://edocs.beasys.co.jp/e-docs/wls/docs70/admin.html`

- テキスト エディタで `config.xml` ファイルを開き、次の要素の資格を変更します。

  ```
  <SecurityConfiguration Credential=”foobar” Name=”mydomain”/>
  ```

  注意： クリア テキスト資格は、最初に `config.xml` ファイルが生成されたときに暗号化されます。

新しい SerializedSystemIni.dat ファイルの生成

WebLogic Integration 7.0 サーバを開始したときに次のような例外が発生した場合、`SerializedSystemIni.dat` ファイルを再生成する必要があります。

```
weblogic.security.internal.FileUtilsException: Couldn't
rename DOMAIN_HOME\SerializedSystemIni.dat to
DOMAIN_HOME\SerializedSystemIni.dat42698.old
```

これらのメッセージでは *DOMAIN_HOME* は 2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した新しいドメインのパス名を表します。

ドメインに新しい SerializedSystemIni.dat ファイルを生成するには、次の手順を実行します。

1. ドメインの存在するディレクトリに移動します。

   ```
   cd DOMAIN_HOME
   ```

   DOMAIN_HOME は 2-6 ページの「手順 5. 新しいドメインの作成およびデータベース情報のコンフィグレーション」で作成した新しいドメインのパス名を表します。

2. config.xml ファイルを開き、Credential 属性から暗号化されたセクションを削除します。たとえばコード リスト 3-6 に表示されているコンフィグレーション セクションを削除し、コード リスト 3-7 に表示されているコンフィグレーション セクションに置き換えます。

コード リスト 3-6 暗号化された Credential 要素

```
<EmbeddedLDAP
    Credential="{3DES}CUnX5jDsmxluwzO45cb+kErCo+3pa92oXcPZl8L23pwx"
Name="mydomain"/>
    <SecurityConfiguration Credential="{3DES}aKaA3CEY4TIx" Name="mydomain"/>
```

コード リスト 3-7 Credential 要素から削除された暗号化箇所

```
<EmbeddedLDAP
    Credential="" Name="mydomain"/>
    <SecurityConfiguration Credential="" Name="mydomain"/>
```

3. fileRealm.properties ファイルのコピーを作成し、プラットフォームに合わせて適切なコマンドを実行してコピー fileRealm.properties.src を呼び出します。

- Windows:

  ```
  copy fileRealm.properties fileRealm.properties.src
  ```

- UNIX:

  ```
  cp fileRealm.properties fileRealm.properties.src
  ```

4. プラットフォームに合わせて適切なコマンドを実行して boot.properties ファイルを削除します。

- Windows:

  ```
  delete boot.properties
  ```

- UNIX:

  ```
  rm boot.properties
  ```

5. プラットフォームに合わせて適切なコマンドを実行して SerializedSystemIni.dat ファイルの名前を変更します。

- Windows:

  ```
  rename SerializedSystemIni.dat SerializedSystemIni.dat.old
  ```

- UNIX:

  ```
  mv SerializedSystemIni.dat SerializedSystemIni.dat.old
  ```

6. fileRealm.properties.src ファイルを開き、すべてのハッシュ パスワードをクリア テキスト パスワードに変更します（パスワードは user.*user_name* 要素によって設定されます。*user_name* は次のリストの一部に太字表示されているように、ユーザ ID を表す）。たとえばコード リスト 3-8 で暗号化されたパスワードを削除し、コード リスト 3-9 に示すように、テキスト バージョンに置き換えます。

コード リスト 3-8 暗号化されたパスワードを持つ fileRealm.properties ファイルのリストの一部

```
group.ConfigureSystem=admin,joe,mary,guest,wlisystem
acl.lookup.weblogic.jndi.weblogic.fileSystem=everyone
acl.enablermonitor.WLCAdmin=admin
user.joe=0xa078cb45e6f6c4eefdd1f14495ff739b5536904c
group.DeleteTemplate=admin,joe,mary,guest,wlisystem
group.AdministerUser=admin,joe,mary,guest,wlisystem
group.Deployers=Administrators
```

コード リスト 3-9 クリア テキスト のパスワードを持つ fileRealm.properties ファイルのリストの一部

```
group.ConfigureSystem=admin,joe,mary,guest,wlisystem
acl.lookup.weblogic.jndi.weblogic.fileSystem=everyone
acl.enablermonitor.WLCAdmin=admin
user.joe=password
group.DeleteTemplate=admin,joe,mary,guest,wlisystem
group.AdministerUser=admin,joe,mary,guest,wlisystem
group.Deployers=Administrators
```

7. プラットフォームに合わせて適切なコマンドを実行して `SerializedSystemIni.dat` ファイルの名前を変更します。
   - Windows:

     ```
     rename SerializedSystemIni.dat SerializedSystemIni.dat.old
     ```

   - UNIX:

     ```
     mv SerializedSystemIni.dat SerializedSystemIni.dat.old
     ```

8. `FileRealm` ユーティリティを使用して `SerializedSystemIni.dat` ファイルを再生成します。具体的には、次のコマンドを実行します。

   ```
   java weblogic.security.internal.acl.FileRealm
   fileRealm.properties SerializedSystemIni.dat
   ```

RosettaNet スキーマの変更点

WebLogic Integration では RosettaNet メッセージのスキーマ検証が提供されています。WebLogic Integration 2.1 および WebLogic Integration 2.1 SP1 の検証のスキーマは、W3C (World Wide Web Consortium) XSD (2000 XML Schema Definitions) スキーマに基づきます。WebLogic Integration 7.0 の検証のスキーマは、W3C (World Wide Web Consortium) XSD (2001 XML Schema Definitions) スキーマに基づきます。

RosettaNet メッセージは 2 つの部分で構成されています。メッセージ ヘッダ XML ドキュメント（サービス - ヘッダ）および XML ビジネス ドキュメントまたはペイロードです。メッセージ ヘッダ XML ドキュメントおよび XML ビジネ

スドキュメントの両方を検証するために使用する XSD スキーマは、W3C 2001 XSD スキーマを使用するようにコンバートされました。WebLogic Integration メッセージの検証のためのスキーマを開発した場合、2001 XSD スキーマを使用するように更新する必要があります（スキーマは、コード リスト 3-11 またはコード リスト 3-12 で示すスキーマ要素で始まる必要がある）。WebLogic Integration 7.0 は、2001 XSD スキーマの RosettaNet メッセージの検証のみサポートしています。

RosettaNet メッセージの検証スキーマは、*WLI_HOME*\lib\xmlschema\rosettanet ディレクトリにあります（*WLI_HOME* は WebLogic Integration ホーム ディレクトリを表す）。

2000 XSD スキーマは、次のリストのスキーマ要素から始まります。

コード リスト 3-10 2000 XSD スキーマ要素

```
<schema xmlns="http://www.w3.org/2000/10/XMLSchema">
       <!-- schema content goes here -->
</schema>
```

2001 XSD スキーマは、コード リスト 3-11 またはコード リスト 3-12 に示すスキーマ要素に類似したものから始まります。コード リスト 3-12 の XSD スキーマ要素は、XSD スキーマの要素を完全に修飾するためにネームスペース xsd を使用します。

コード リスト 3-11 2001 XSD スキーマ要素

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
       <!-- schema content goes here -->
</schema>
```

コード リスト 3-12 xsd ネームスペースの付いた 2001 XSD スキーマ要素

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
       <!-- schema content goes here -->
</schema>
```

xml: ネームスペース

WebLogic Integration 7.0 では xml: ネームスペースの使用をサポートしています。xml: ネームスペースは、次の属性名を含む標準 W3C ネームスペースです。

- xml:lang －使用される自然言語を識別します。
- xml:space －空白が要素にとって重要であるかを指定します。
- xml:base － XML 文書の部分のベース URI を定義します。

WebLogic Integration 2.1 または 2.1 SP1 XSD スキーマ ファイルでは、次のように、xml:lang は文字列にコンバートされます。

```
<xsd:attribute name="xml:lang" type="xsd:string"/>
```

WebLogic Integration 7.0 XSD スキーマ ファイルでは、xml:lang の値は xml: ネームスペース によって決定されます。メッセージの検証のために独自のスキーマを開発した場合、次のリストに示すように、xml:lang 属性を変更してください。

```
<xsd:attribute ref="xml:lang"/>
```

デフォルトでは WebLogic Integration 7.0 は、xml: ネームスペース XSD スキーマを次の標準 xml: ネームスペース URL でからではなく、ローカル ファイルシステムから取得します。

```
http://www.w3.org/2001/xml.xsd
```

ローカル ファイルシステムでネームスペースをルックアップすることにより、パフォーマンスが向上し、ネットワーク トラフィックが軽減されます。

URL からではなくローカル ファイルシステムからスキーマを取得することにより、WebLogic Integration 7.0 ではネットワーク トラフィックを軽減してパフォーマンスを向上させることができます。たとえば、次のリストに示すように、0A1_MS_V02_00_FailureNotification.xsd スキーマによって schemaLocation 属性値を schemas/xml.xsd に設定します。この設定により、xml: ネームスペース スキーマは、ローカル ファイルシステムから取得されます。

コード リスト 3-13 ローカル スキーマを使用した 0A1_MS_V02_00_FailureNotification.xsd

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
<xsd:import namespace = "http://www.w3.org/XML/1998/namespace"
schemaLocation = "schemas/xml.xsd"/>
.
.
.
```

コード リスト 3-14 では、schemaLocation 属性の値は http://www.w3.org/2001/xml.xsd です。この設定が使用された場合、WebLogic Integration では、URL を使用して xml: ネームスペース スキーマが取得されます。

コード リスト 3-14 URL スキーマ を使用した 0A1_MS_V02_00_FailureNotification.xsd

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
<xsd:import namespace = "http://www.w3.org/XML/1998/namespace"
schemaLocation = "http://www.w3.org/2001/xml.xsd"/>
.
.
.
```

URL から xml: ネームスペースを検証する場合、*WLI_HOME*\lib\xmlschema\rosettanet ディレクトリの XSD スキーマ ファイルを標準の xml: ネームスペース URL に schemaLocation 属性の値を設定する XSD スキーマ ファイル セットに置き換えることができます。手順は、次のとおりです。

1. プラットフォームに合わせて適切なコマンドを実行して、RosettaNet XSD スキーマ ディレクトリに移動します。
   - Windows:

     ```
     cd WLI_HOME\lib\xmlschema\rosettanet
     ```

- UNIX:

  ```
  cd WLI_HOME/lib/xmlschema/rosettanet
  ```

いずれのコマンド ラインの場合でも、*WLI_HOME* は、WebLogic Integration ホーム ディレクトリを表します。

2. プラットフォームに合わせて適切なコマンドを実行して、バックアップ ディレクトリを作成します。
   - Windows:

     ```
     md backup
     ```

   - UNIX:

     ```
     mkdir backup
     ```

3. プラットフォームに合わせて適切なコマンドを実行して、ローカル `xml:` ネームスキーマ ファイルにポイントする XSD スキーマ ファイルをバックアップ ディレクトリに移動します。
   - Windows:

     ```
     move *.xsd backup
     ```

   - UNIX:

     ```
     mv *.xsd backup
     ```

4. 標準の `xml:` ネームスペース URL にポイントする XSD スキーマ ファイルを抽出します。

   ```
   jar xvf schemas.jar
   ```

追加の 2001 XSD スキーマの変更点

独自のメッセージ検証のスキーマを開発する場合、スキーマに次の変更を加える必要がある場合があります。

- `appInfo` 要素を `documentation` 要素に置き換えます。この置き換えは DTD グラマーでの変更に対応するために WebLogic Integration 7.0 スキーマで行われました。
- 次の例のリストのとおり、`content=" elementOnly"` および `content=" textOnly"` 属性を削除します。

コード リスト 3-15 2000 XSD スキーマの例（WebLogic Integration 2.1 または 2.1 SP1）

```
<xsd:complexType content = "elementOnly">
<xsd:choice>
       <xsd:element ref = "GlobalLocationIdentifier"/>
       <xsd:group ref = "StreetAddress"/>
       <xsd:group ref = "GlobalLocationIdAndStreetAddress"/>
</xsd:choice>
```

コード リスト 3-16 2001 XSD スキーマの例（WebLogic Integration 7.0）

```
<xsd:complexType>
<xsd:choice>
       <xsd:element ref = "GlobalLocationIdentifier"/>
       <xsd:group ref = "StreetAddress"/>
       <xsd:group ref = "GlobalLocationIdAndStreetAddress"/>
</xsd:choice>
```