Oracle® Application Server

Adapter 概要

10*g*（10.1.3.1.0）

部品番号 : B31900-01

2006 年 12 月

ORACLE®

Oracle Application Server Adapter 概要 , 10*g*（10.1.3.1.0）

部品番号 : B31900-01

原本名 : Oracle Application Server Adapter Concepts Guide, 10*g* Release 3 (10.1.3.1.0)

原本部品番号 : B31005-01

原本著者 : Sheela Vasudevan

原本協力者 : Bo Stern, Meera Srinivasan, Maneesh Joshi, Prasad Dixit-Hardikar

目次

4 レガシー・アダプタ

5 アダプタと Oracle Application Server コンポーネントとの統合

6 アダプタのライフ・サイクル管理

はじめに

ここでは、次の項目について説明します。

対象読者

このマニュアルは、次のアダプタの概念について学習するユーザーを対象としています。

- テクノロジ
- パッケージ・アプリケーション
- レガシー

ドキュメントのアクセシビリティについて

オラクル社は、障害のあるお客様にもオラクル社の製品、サービスおよびサポート・ドキュメントを簡単にご利用いただけることを目標としています。オラクル社のドキュメントには、ユーザーが障害支援技術を使用して情報を利用できる機能が組み込まれています。HTML 形式のドキュメントで用意されており、障害のあるお客様が簡単にアクセスできるようにマークアップされています。標準規格は改善されつつあります。オラクル社はドキュメントをすべてのお客様がご利用できるように、市場をリードする他の技術ベンダーと積極的に連携して技術的な問題に対応しています。オラクル社のアクセシビリティについての詳細情報は、Oracle Accessibility Program の Web サイト `http://www.oracle.com/accessibility/` を参照してください。

ドキュメント内のサンプル・コードのアクセシビリティについて

JAWS（Windows のスクリーン・リーダー）は、ドキュメント内のサンプル・コードを正確に読めない場合があります。コード表記規則では閉じ括弧だけを行に記述する必要があります。しかし JAWS は括弧だけの行を読まない場合があります。

外部 Web サイトのドキュメントのアクセシビリティについて

このドキュメントにはオラクル社およびその関連会社が所有または管理しない Web サイトへのリンクが含まれている場合があります。オラクル社およびその関連会社は、それらの Web サイトのアクセシビリティに関しての評価や言及は行っておりません。

Oracle サポート・サービスへの TTY アクセス

アメリカ国内では、Oracle サポート・サービスへ 24 時間年中無休でテキスト電話（TTY）アクセスが提供されています。TTY サポートについては、(800)446-2398 にお電話ください。

関連ドキュメント

詳細は、次のドキュメントを参照してください。

- 『Oracle Application Server 管理者ガイド』
- 『Oracle Application Server Adapters for Files, FTP, Databases および Enterprise Messaging ユーザーズ・ガイド』

表記規則

このマニュアルの本文では、次の表記規則を使用します。

規則	意味
太字	太字は、アクションに関連付けられているグラフィカル・ユーザー・インタフェース要素と、本文中または用語集で定義されている用語を示します。
イタリック体	イタリック体は、特定の値を指定する必要があるプレースホルダや変数を示します。
固定幅フォント	固定幅フォントは、段落内のコマンド、URL、サンプル・コード、画面に表示されるテキスト、または入力テキストを示します。

サポートおよびサービス

次の各項に、各サービスに接続するための URL を記載します。

Oracle サポート・サービス

オラクル製品サポートの購入方法、および Oracle サポート・サービスへの連絡方法の詳細は、次の URL を参照してください。

```
http://www.oracle.co.jp/support/
```

製品マニュアル

製品のマニュアルは、次の URL にあります。

```
http://otn.oracle.co.jp/document/
```

研修およびトレーニング

研修に関する情報とスケジュールは、次の URL で入手できます。

```
http://www.oracle.co.jp/education/
```

その他の情報

オラクル製品やサービスに関するその他の情報については、次の URL から参照してください。

```
http://www.oracle.co.jp
http://otn.oracle.co.jp
```

注意： ドキュメント内に記載されている URL や参照ドキュメントには、Oracle Corporation が提供する英語の情報も含まれています。日本語版の情報については、前述の URL を参照してください。

1

概要

ビジネス・プロセスの最適化の必要性が高まるにつれて、既存のバックエンド・アプリケーションといかに効率的に統合するかということが、成功の鍵となってきています。ビジネス・プロセスを最適化するために、Oracle Application Server アダプタを使用してアプリケーションを統合できます。アダプタによって堅牢かつ軽量で高スケーラブルな標準ベースの統合フレームワークがサポートされ、種類の異なったアプリケーション間の相互通信が可能になります。たとえば、アダプタを使用すると、パッケージ・アプリケーション、レガシー・アプリケーション、データベースおよび Web サービスを統合できます。様々なベンダーにより提供される、異なった技術に基づく、異なったプラットフォームで稼働する異機種アプリケーションを統合することによって、相互運用性を確保できます。

注意： このマニュアルでは、アダプタと、OC4J、Business Process Execution Language for Web Services（BPEL）Process Manager、OracleAS Integration InterConnect および Oracle Enterprise Service Bus との統合について説明します。

この章の内容は、次のとおりです。

1.1 Oracle Application Server アダプタの機能

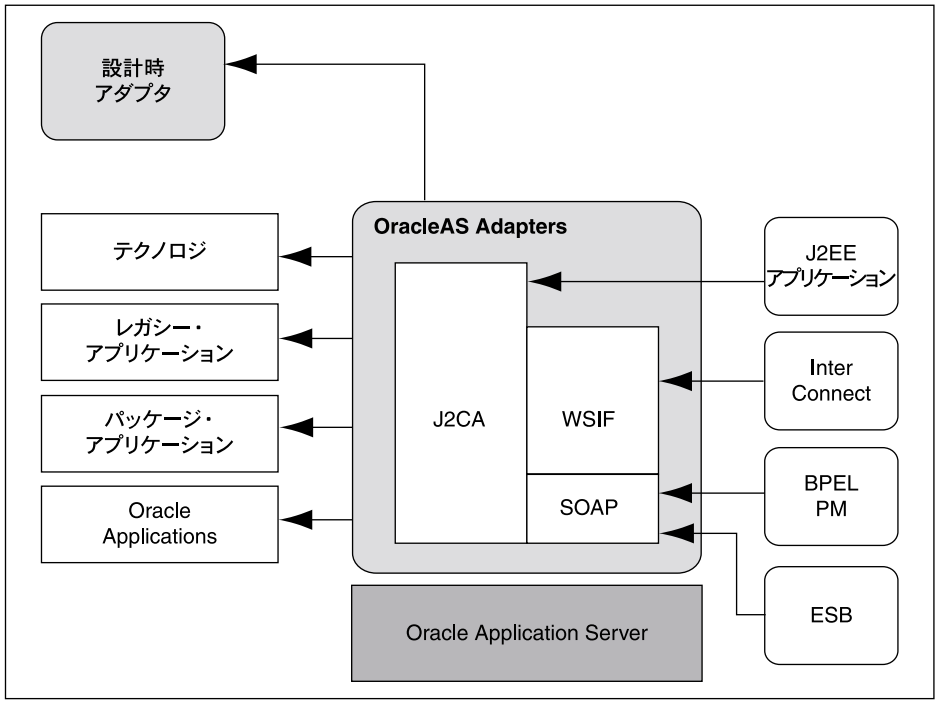
Oracle Application Server アダプタには、次の利点があります。

- 複雑なビジネス・プロセスを統合するための接続プラットフォームの提供 : アダプタは、メインフレーム・アプリケーションおよびレガシー・アプリケーションを、Enterprise Resource Planning（ERP）、カスタマ・リレーションシップ・マネジメント（CRM）、データベースおよびメッセージ・システムと統合します。Oracle には、SAP や Siebel などの様々なパッケージ・アプリケーション、およびデータベースに接続するためのアダプタが 200 以上用意されています。また、アダプタは、MQSeries や Oracle Advanced Queuing などのミドルウェア・メッセージ・システムと Tuxedo などのレガシー・アプリケーションを統合して、完全なソリューションを提供します。
- オープン標準のサポート : アダプタは、J2EE Connector Architecture（J2CA）、Extensible Markup Language（XML）、Web Service Invocation Framework（WSIF）、Web Service Inspection Language（WSIL）および Web Service Definition Language（WSDL）などの標準セットに基づいています。標準をサポートすることによって、ユーザーの習熟曲線が短くなり、ユーザーが単一のベンダーに依存する割合が低くなります。
- サービス指向アーキテクチャ（SOA）の実装 : オープン標準をサポートすることによって、アダプタは、疎結合、柔軟性および拡張性を容易にする SOA を実装します。
- ネイティブ API の使用 : アダプタは、バックエンド・システムとインタフェースする方法を複数サポートし、様々なデプロイ・オプションを提供します。ネイティブ API を使用して、アダプタはバックエンド・アプリケーションと通信し、ネイティブ・データを、クライアントに提供する標準 XML に変換します。
- データのモデル化 : アダプタは、設計時に構成されるアダプタ・メタデータに基づいて、ネイティブ API と標準 XML 間の変換を実行します。アダプタ構成は設計時に定義され、ランタイム・コンポーネントで使用されています。
- リアルタイム接続性および双方向接続性の簡略化 : アダプタは、様々なバックエンド・システムとの双方向通信を提供します。双方向通信では、バックエンド・システムへのリクエストの送信、およびレスポンスの受信が行われます。また、アダプタはリアルタイムのイベント通知サービスもサポートします。このサービスでは、バックエンド・データの作成、削除および更新に関する正常なバックエンド・トランザクションに関連付けられたバックエンド・イベントについて通知されます。この双方向接続性によって、迅速で柔軟性のある効率的な統合が保証され、統合のコストが削減されます。
- 可用性の最大化 : アダプタは、J2CA 1.5 仕様に準拠し、Oracle Application Server の J2CA コンテナである Oracle Containers for J2EE（OC4J）にデプロイされます。したがって、基礎となる Oracle Application Server プラットフォームのスケーラビリティおよび高可用性を最大限利用できます。また、アダプタは JBoss および Weblogic プラットフォームでデプロイできます。
- 使いやすい設計時ツールの提供 : アダプタは、アダプタを構成および管理するための Graphical User Interface（GUI）を提供する設計時ツールを使用して、迅速な実装とデプロイを実現します。また、ツールを使用すると、バックエンド・スキーマを参照、ダウンロードおよび構成できます。
- Oracle Application Server コンポーネントとのシームレスな統合のサポート : アダプタは、Oracle Application Server Portal などの Oracle Application Server コンポーネント、BPEL Process Manager、および Enterprise JavaBeans（EJB）、サーブレット、Java アプリケーションなどの J2EE アプリケーションと統合されます。

1.2 Oracle Application Server アダプタのタイプ

Oracle Application Server アダプタには、テクノロジ、パッケージ・アプリケーションおよびレガシーの 3 つのタイプがあります。図 1-1 は、様々なタイプのアダプタを図で説明しています。

図 1-1 Oracle Application Server アダプタのタイプ



この項では、次のタイプのアダプタについて説明します。

- テクノロジ・アダプタ
- パッケージ・アプリケーション・アダプタ
- レガシー・アダプタ

1.2.1 テクノロジ・アダプタ

テクノロジ・アダプタは、Oracle Application Server を、トランスポート・プロトコル、データ・ストアおよびメッセージ・ミドルウェアと統合します。これらのアダプタには、OracleAS Adapter for FTP、OracleAS Adapter for JMS、OracleAS Adapter for Database、OracleAS Adapter for Advanced Queuing および OracleAS Adapter for MQ Series があります。テクノロジ・アダプタは現在、BPEL Process Manager インストレーションで使用できます。

1.2.2 パッケージ・アプリケーション・アダプタ

パッケージ・アプリケーション・アダプタは、Oracle Application Server を、SAP や Siebel などの様々なパッケージ・アプリケーションと統合します。これらのアダプタには、OracleAS Adapter for Oracle Applications、OracleAS Adapter for PeopleSoft、OracleAS Adapter for SAP R/3、OracleAS Adapter for Siebel および OracleAS Adapter for J.D. Edwards があります。パッケージ・アプリケーション・アダプタは、OracleAS Adapters CD の一部として使用できます。

1.2.3 レガシー・アダプタ

レガシー・アダプタは、Oracle Application Server を、レガシー・アプリケーションおよびメインフレーム・アプリケーションと統合します。これらのアダプタには、OracleAS Adapter for Tuxedo、OracleAS Adapter for CICS、OracleAS Adapter for VSAM、OracleAS Adapter for IMS/TM および OracleAS Adapter for IMS/DB があります。レガシー・アダプタは、OracleAS Adapters CD の一部として使用できます。

1.3 Oracle Application Server アダプタ・サービスのタイプ

アダプタには、アプリケーション間の通信を容易にするために次のタイプのサービスが用意されています。

- リクエスト / レスポンス（アウトバウンド相互作用）サービス
- イベント通知（インバウンド相互作用）サービス
- メタデータ・サービス

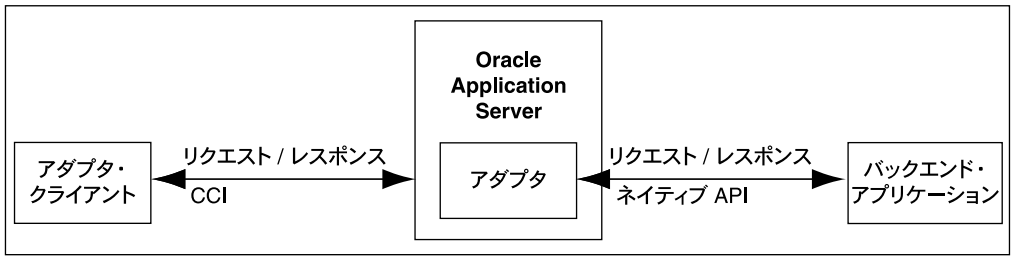
1.3.1 リクエスト / レスポンス（アウトバウンド相互作用）サービス

アダプタは、同期リクエスト / レスポンス・サービスをサポートしています。アダプタは、アダプタ・クライアントからリクエストを受信し、これらのリクエストをバックエンドのネイティブ・データ形式に変換し、バックエンド・アプリケーションで適切なメソッドをコールします。さらに、リクエスト / レスポンス・サービスは、リバース・トランスレーションの実行後、アダプタ・フレームワーク・コンポーネントに対するバックエンド・レスポンスを取得します。J2CA 用語では、このタイプのサービスは、アウトバウンド相互作用とも呼ばれます。

リクエスト / レスポンス・サービスは、バックエンド・データの作成、削除、更新および問合せに使用でき、バックエンド・ワークフローおよびトランザクションのコールにも使用できます。たとえば、OC4J アプリケーション・クライアントは、OracleAS Adapter for SAP を使用して、SAP アプリケーション内でカスタマを作成できます。

図 1-2 は、リクエスト / レスポンス・サービスを図で説明しています。

図 1-2 リクエスト / レスポンス・サービス



1.3.2 イベント通知（インバウンド相互作用）サービス

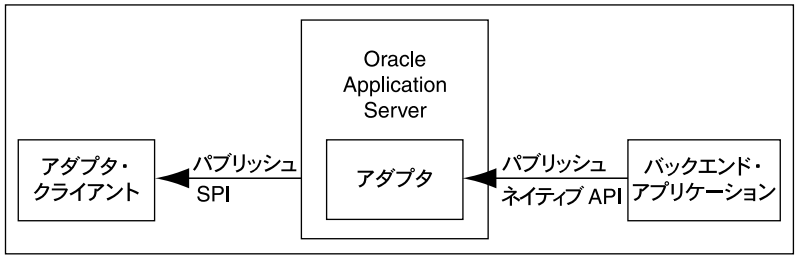
アダプタは、非同期通信パラダイムであるイベント通知サービスをサポートしています。J2CA用語では、このタイプのサービスは、インバウンド相互作用とも呼ばれます。

アダプタは、バックエンド・イベントの変更をリスニングまたはポーリングします。イベントをリスニングしているとき、アダプタは、イベントをアダプタに送信するように構成されたバックエンド・アプリケーションのリスナーとして登録されています。また、クライアント・アプリケーションが要求したイベントに対して、バックエンド・アプリケーション（通常はデータベースまたはファイル）をポーリングすることもできます。

イベント通知サービスは、バックエンド・データの作成、削除および更新に関する正常なバックエンド・トランザクションに関連付けられたバックエンド・イベントの追跡に使用できます。

図 1-3 は、イベント通知サービスを図で説明しています。

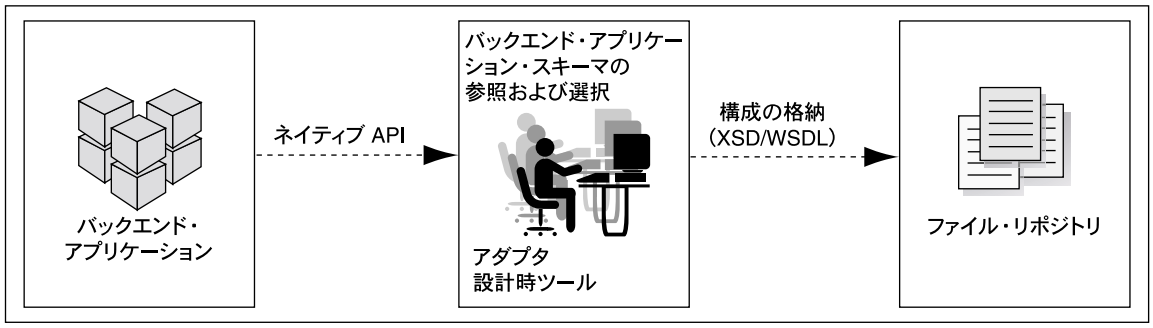
図 1-3 イベント通知サービス



1.3.3 メタデータ・サービス

アダプタ・メタデータ定義には、バックエンド接続、およびビジネス・オブジェクトとビジネス・サービスのスキーマに関する情報が格納されています。アダプタは、メタデータを参照および格納するための設計時コンポーネントと、サービスを実行するためのランタイム・コンポーネントで構成されています。アダプタ・メタデータ定義は、XML スキーマ定義（XSD）および WSDL ファイルとして生成されます。図 1-4 は、メタデータ相互作用を図で説明しています。

図 1-4 メタデータ・サービス



2

テクノロジ・アダプタ

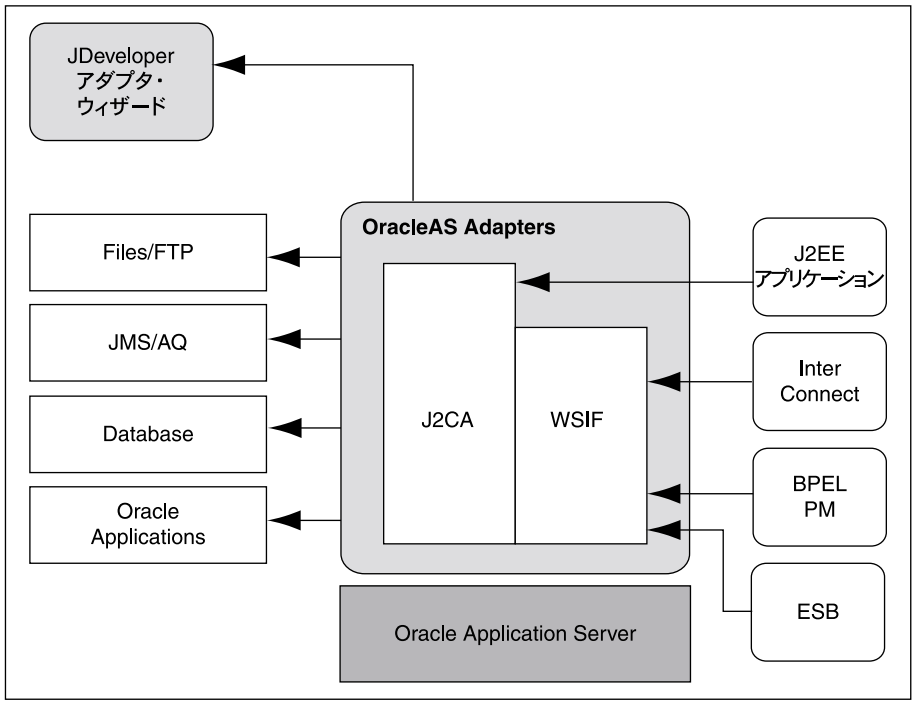
テクノロジ・アダプタは、Oracle Application Server を、トランスポート・プロトコル、データ・ストアおよびメッセージ・ミドルウェアと統合します。これらのアダプタには、OracleAS Adapter for FTP、OracleAS Adapter for JMS、OracleAS Adapter for Databases、OracleAS Adapter for Advanced Queuing および OracleAS Adapter for MQ Series があります。

この章の内容は、次のとおりです。

2.1 アーキテクチャ

テクノロジ・アダプタは J2EE Connector Architecture（J2CA）1.5 標準に準拠しており、BPEL Process Manager と同じ Oracle Containers for J2EE（OC4J）にリソース・アダプタとしてデプロイされます。OracleAS Adapter for Oracle Applications は、テクノロジ・アダプタと同じアーキテクチャで構成されています。図 2-1 は、テクノロジ・アダプタのアーキテクチャを図で説明しています。

図 2-1 テクノロジ・アダプタのアーキテクチャ



2.2 設計時コンポーネント

設計時に、テクノロジ・アダプタは、Oracle JDeveloper を使用してアダプタ・メタデータを生成します。リクエスト / レスポンス・サービス（J2CA アウトバウンド相互作用とも呼ばれます）とイベント通知サービス（J2CA インバウンド相互作用とも呼ばれます）は、J2CA WSIF WSDL ファイルに記述されます。これらの WSDL ファイルは J2CA 拡張要素で構成されます。J2CA WSDL 要素は、J2CA 1.5 リソース・アダプタを BPEL Process Manager とシームレスに統合するためにアダプタ・フレームワークによって使用されます。

テクノロジ・アダプタと BPEL Process Manager の統合の詳細は、5.2 項「アダプタと BPEL Process Manager の統合」を参照してください。

例 2-1 OracleAS Adapter for Databases 用の WSDL ファイルの生成

OracleAS Adapter for Databases は、JDeveloper を使用して構成できます。このアダプタは、データ操作の実行、ストアド・プロシージャまたはファンクションのコール、およびデータベース・イベントのリアルタイムのパブリッシュに利用できます。アダプタ定義を構成するには、図 2-2 に示すように、「アダプタ・タイプ」ダイアログ・ボックスから**「データベース・アダプタ」**を選択します。

図 2-2 データベース・アダプタの選択

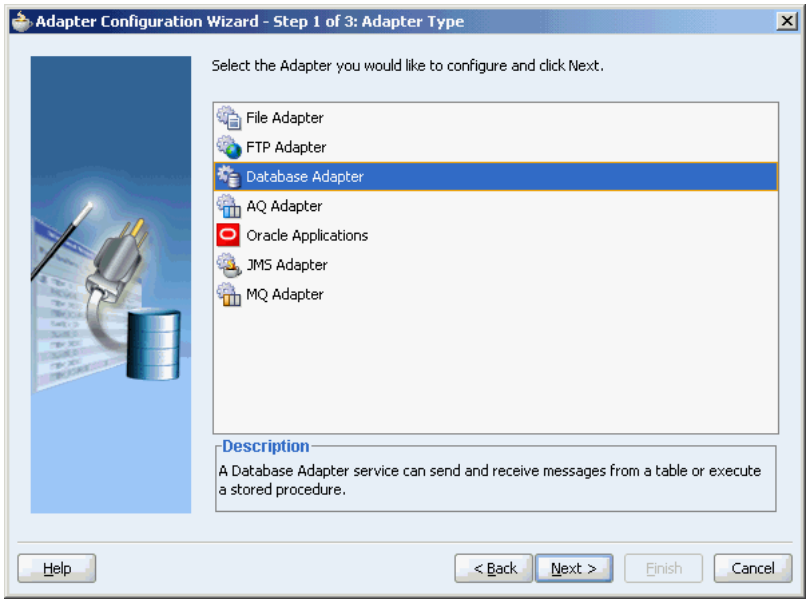


図 2-3 は、「表のインポート」ウィンドウを参照して、アダプタに必要な表を選択する方法を示しています。

図 2-3 必要な表の参照

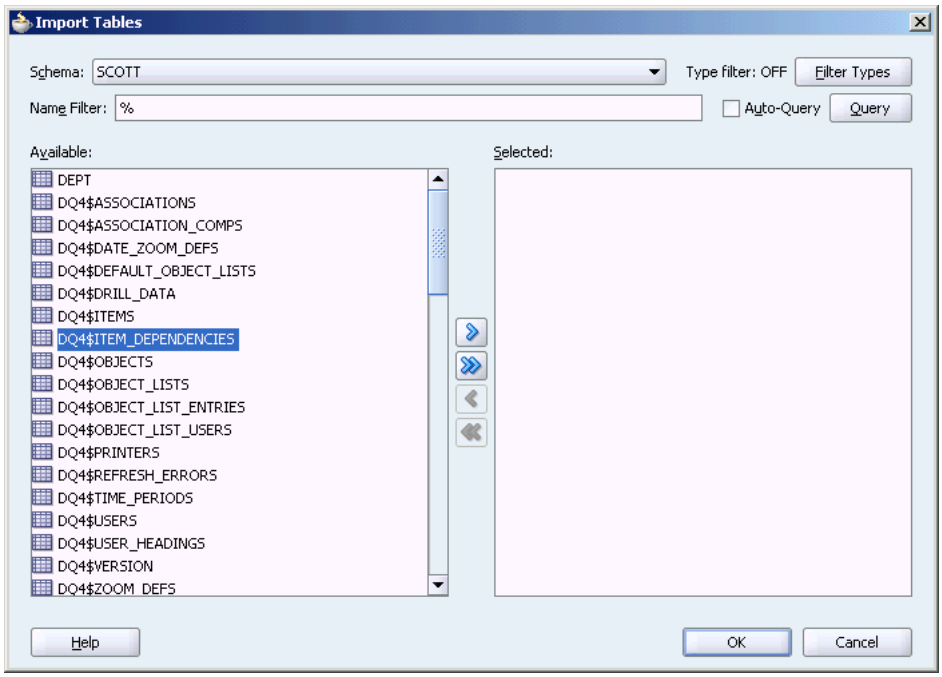
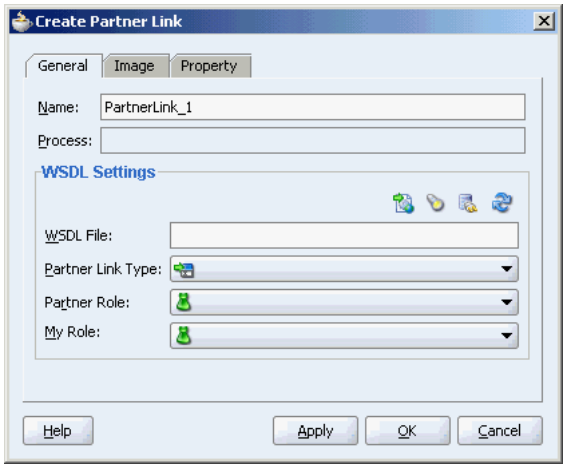


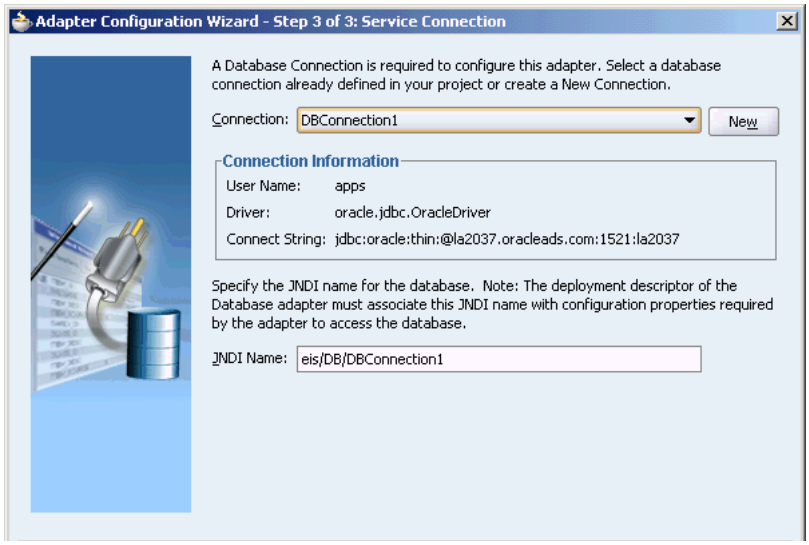
図 2-4 は、OracleAS Adapter for Databases 用の WSDL 設定の指定方法を示しています。

図 2-4 WSDL 設定の指定



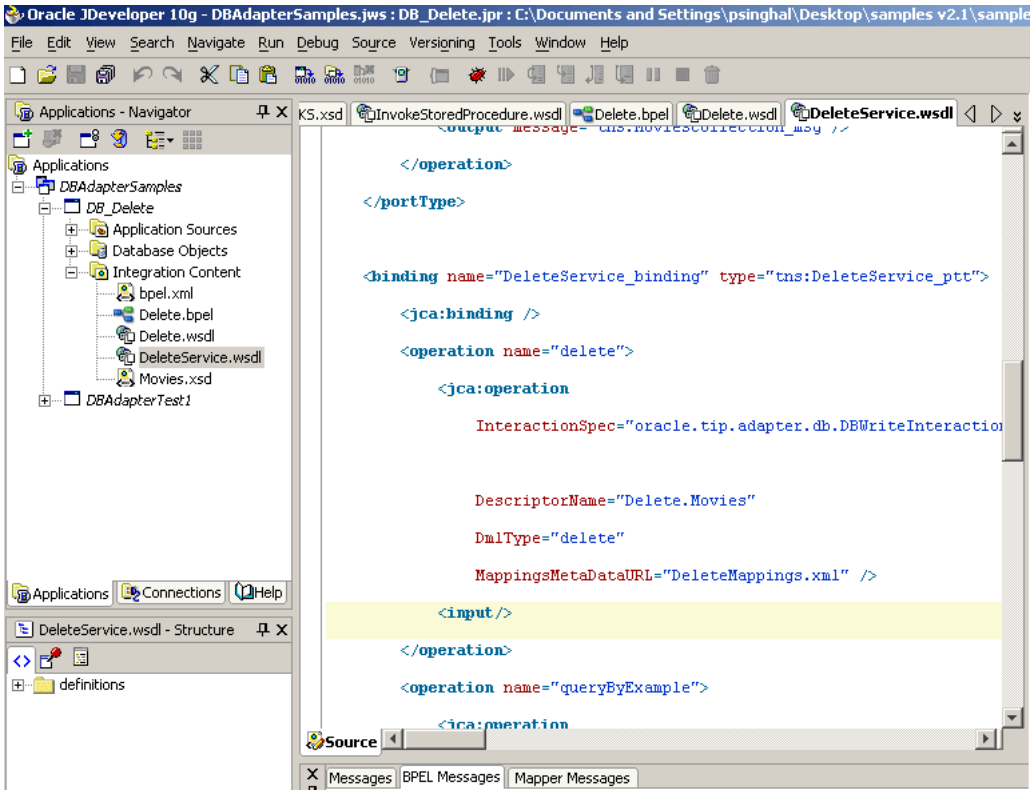
次に、データベース接続を確立し、操作タイプを選択して、必要な表を選択する必要があります。ランタイム接続パラメータは、oc4j-ra.xml ファイルで指定され、設計時に指定される Java Naming and Directory Interface（JNDI）名にリンクされます。図 2-5 は、新規データベース接続の作成を示しています。

図 2-5 新規データベース接続の作成



最後に、JDeveloper によって、図 2-6 に示すような、OracleAS Adapter for Databases 用の J2CA バインドが指定された WSDL ファイルが生成されます。

図 2-6 WSDL ファイルの構造



注意： 次の例に示すように、アクティブ化エージェント下で `bpel.xml` に設定を追加することで、テクノロジ・アダプタが大規模な XML ペイロードを処理できるようすることが可能です。

```
<activationAgents>
   <activationAgent ...>
      ...
   <property name="postAsString">true</property>
     ...
```

2.3 ランタイム・コンポーネント

テクノロジ・アダプタのランタイム・コンポーネントは、特定のバックエンド・アプリケーションに対する J2CA 1.5 リソース・アダプタです。テクノロジ・アダプタは、Oracle Application Server の J2CA コンテナである OC4J にデプロイされます。BPEL Process Manager は、Web サービス・メッセージと J2CA 相互作用の間の変換を実行するアダプタ・フレームワークを介して、これらの J2CA 1.5 アダプタと統合されます。

BPEL Process Manager はアダプタ・フレームワークを使用して、リクエスト / レスポンス・サービス（J2CA アウトバウンド相互作用）を BPEL の invoke アクティビティに統合し、アダプタ・イベントを BPEL の receive アクティビティにパブリッシュします。BPEL Process Manager との統合の詳細は、5.2 項「アダプタと BPEL Process Manager の統合」を参照してください。

注意： テクノロジ・アダプタと統合可能なのは、Business Process Execution Language for Web Services（BPEL）Process Manager のみです。

2.4 デプロイ

テクノロジ・アダプタは、インストール時に、BPEL Process Manager と同じ OC4J コンテナ内に J2CA 1.5 リソース・アダプタとしてデプロイされます。テクノロジ・アダプタは物理的に J2CA 1.5 リソース・アダプタとしてデプロイされますが、これらの論理デプロイには、`oc4j-ra.xml` ファイルの編集と設計時の JDeveloper の使用による J2CA 1.5 リソース・アダプタ用のコネクション・ファクトリ・エントリの作成が含まれます。JDeveloper を使用して、JNDI 名を指定します。この名前は、BPEL サーバーへのサービスのデプロイ時に使用される接続のためのプレースホルダとして機能します。この指定によって、開発とその後の本番環境で異なるデータベースを使用できます。ただし、論理デプロイの変更を有効にするには、OC4J コンテナ・プロセスを再起動する必要があります。

3

パッケージ・アプリケーション・アダプタ

パッケージ・アプリケーション・アダプタを使用すると、Oracle Application Server を、J.D. Edwards や Siebel などのパッケージ・アプリケーションと統合できます。これらのアダプタには、OracleAS Adapter for Peoplesoft、OracleAS Adapter for SAP R/3、OracleAS Adapter for Siebel、OracleAS Adapter for Oracle Applications および OracleAS Adapter for J.D. Edwards があります。

この章では、パッケージ・アプリケーション・アダプタのアーキテクチャについて説明します。内容は次のとおりです。

3.1 アーキテクチャ

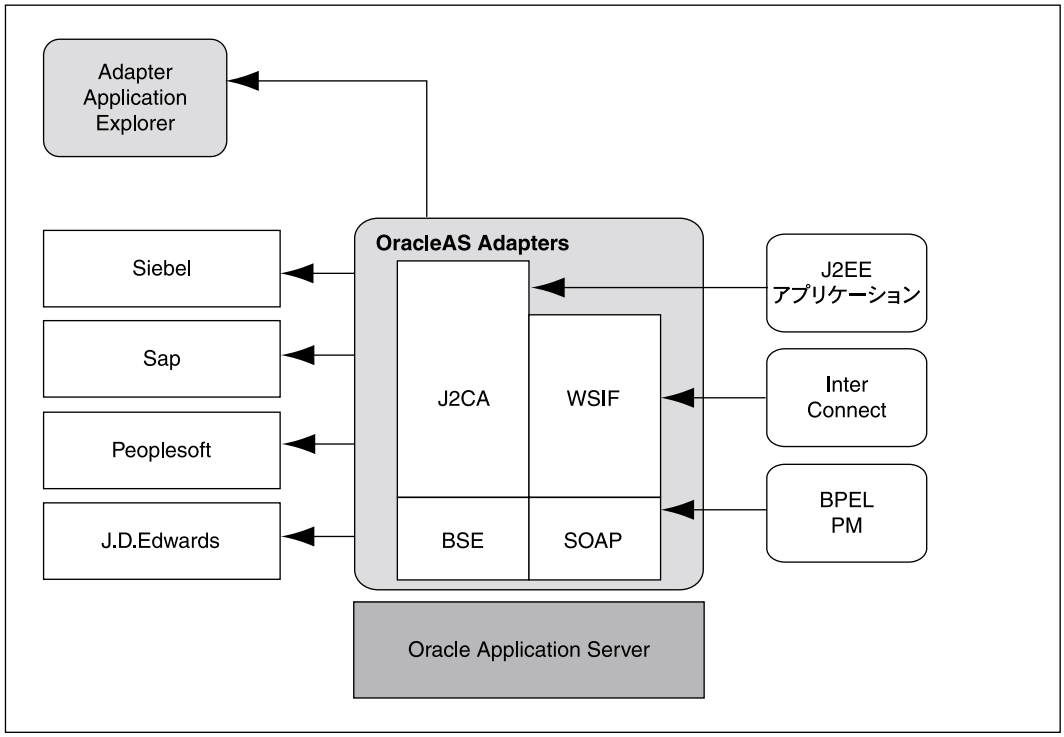
パッケージ・アプリケーション・アダプタは、J2EE Connector Architecture（J2CA）1.5 リソース・アダプタとして、または Web サービス・サーブレットとして Oracle Containers for J2EE（OC4J）コンテナ内にデプロイできます。J2CA インタフェースの他に、パッケージ・アプリケーション・アダプタは、Web Service Definition Language（WSDL）および Simple Object Access Protocol（SOAP）インタフェースをサポートします。パッケージ・アプリケーション・アダプタの J2CA および Web サービスのデプロイには、リポジトリ・プロジェクトが必要です。J2CA のデプロイでは、リソース・アダプタは、複数のバックエンド接続オブジェクトが格納されるリポジトリ・プロジェクトを指します。デプロイメント・ディスクリプタである `oc4j-ra.xml` は、J2CA リポジトリ・プロジェクト、および J2CA リポジトリ・プロジェクト内のアクセスに対する接続名を指します。WSDL のデプロイでは、WSDL リポジトリ・プロジェクトは、アダプタ・メタデータが記述されている WSDL ファイルのセットで構成されます。

> **注意：** このリリースでは、WSDL および SOAP の拡張要素は、OracleAS Adapter for SAP、OracleAS Adapter for Siebel、OracleAS Adapter for Peoplesoft および OracleAS Adapter for J.D. Edwards の 4 つのパッケージ・アプリケーション・アダプタでのみサポートされます。OracleAS Adapter for Oracle Applications のアーキテクチャは、テクノロジ・アダプタと同じです。

パッケージ・アプリケーション・アダプタのアーキテクチャは、OracleAS Adapter Application Explorer（Application Explorer）、J2CA 1.5 リソース・アダプタおよび Business Services Engine（BSE）で構成されます。

図 3-1 は、パッケージ・アプリケーション・アダプタのアーキテクチャを図で説明しています。

図 3-1 パッケージ・アプリケーション・アダプタのアーキテクチャ



この項では、パッケージ・アプリケーション・アダプタのアーキテクチャの次のコンポーネントについて説明します。

- Application Explorer
- BSE
- J2CA 1.5 リソース・アダプタ

3.1.1 Application Explorer

Application Explorer は、パッケージ・アプリケーション・アダプタを構成するための Java Swing ベースの設計時ツールです。Application Explorer を使用すると、バックエンド・アプリケーション接続を構成し、バックエンド・アプリケーション・スキーマを参照して、これらのスキーマをアダプタ・サービスとして公開できます。Application Explorer は、バックエンド・アプリケーション固有のメタデータを参照するためのアプリケーション固有のプラグインとともに提供されます。

Application Explorer を使用すると、OracleAS Adapter の J2CA または BSE 用のリポジトリ・プロジェクトを作成できます。各リポジトリ・プロジェクトは、複数のバックエンド・アプリケーション接続で構成される場合があります。スキーマは、OracleAS Adapter J2CA インタフェース用の XML Schema Definition（XSD）、または SOAP バインドによる WSDL で表されます。

3.1.2 BSE

Application Explorer は、Oracle Application Server の Oracle Containers for J2EE（OC4J）コンテナにデプロイされる BSE とともに動作します。BSE では、クライアントからのリクエストの受入れ、バックエンド・アプリケーションとの相互作用およびバックエンド・アプリケーションからクライアントへのレスポンスの送信について、プロトコルとして SOAP が使用されます。

3.1.3 J2CA 1.5 リソース・アダプタ

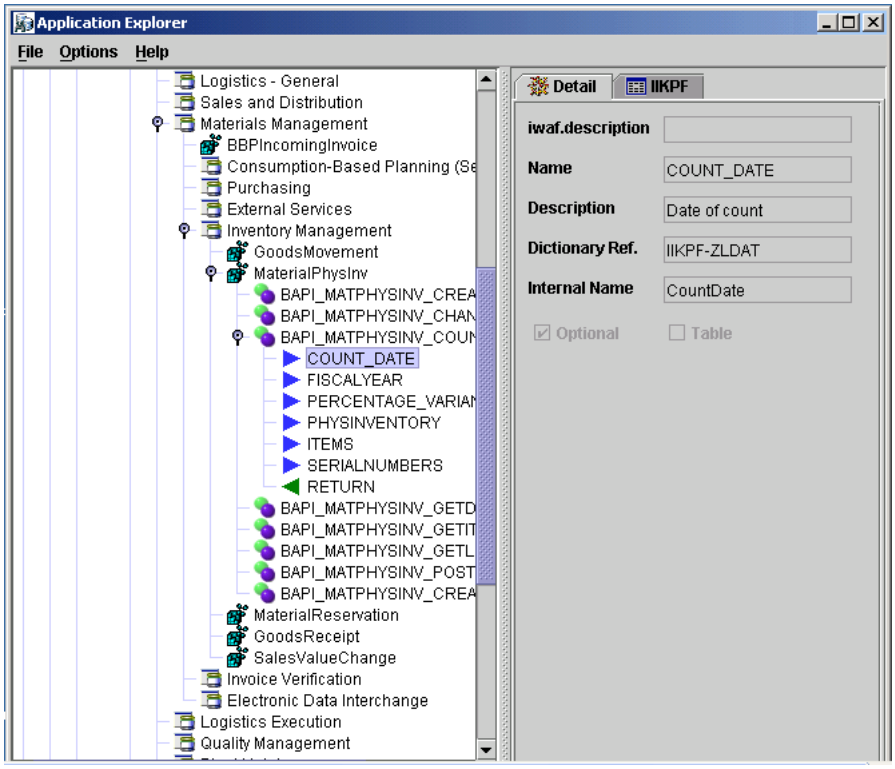
J2CA 1.5 リソース・アダプタは、バックエンド・イベントを受信するためのチャネル・コンポーネントで構成されます。

3.2 設計時コンポーネント

Application Explorer は、設計時にパッケージ・アプリケーション・アダプタの構成に使用されます。このツールを使用して、バックエンド接続のリストが格納される、J2CA 1.5 リソース・アダプタ用のリポジトリ・プロジェクトが作成されます。Application Explorer は、バックエンド・メタデータを XSD、および J2CA 拡張要素が指定された WSDL として公開します。XSD メタデータは、OC4J アプリケーション・クライアントによって、J2CA Common Client Interface（CCI）Application Programming Interface （API）を介した統合に使用されます。J2CA 拡張要素が指定された WSDL は、Business Process Execution Language for Web Services（BPEL）Process Manager との統合に使用されます。BSE メタデータは、WSDL または SOAP として定義できます。

図 3-2 は Application Explorer を示しています。

図 3-2　Application Explorer



例 3-1　OracleAS Adapter for SAP 用の XML リクエスト・スキーマの生成

Application Explorer を使用すると、OracleAS Adapter for SAP 用の接続を確立できます。このためには、次の図に示すように、最初に OracleAS Adapter for SAP に対するターゲットを定義する必要があります。

図 3-3　OracleAS Adapter for SAP の選択

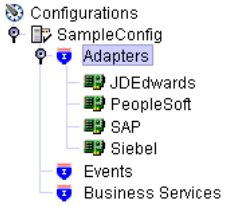
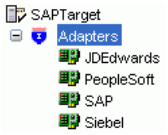
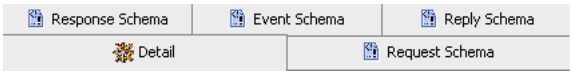


図 3-4 OracleAS Adapter for SAP に対するターゲットの定義



SAP のビジネス機能ライブラリの内容を表示し、オブジェクトを選択した後は、Application Explorer を使用して、その機能に対する XML リクエスト・スキーマと XML レスポンス・スキーマを作成できます。各スキーマ・タイプに対する XML を表示するには、次の図に示すように、必要なタブを選択します。

図 3-5 XML スキーマの表示



3.3 ランタイム・コンポーネント

パッケージ・アプリケーション・アダプタのランタイム・コンポーネントには、J2CA 1.5 リソース・アダプタ、BSE およびサーブレットがあります。OC4J アプリケーション・クライアントは、CCI API を使用して J2CA 1.5 リソース・アダプタと直接インタフェースします。このリリースでは、OC4J は J2CA 1.0 仕様のみをサポートし、CCI API は J2CA アウトバウンド相互作用のコールにのみ使用されます。J2CA 1.5 リソース・アダプタは、アダプタ・フレームワークを介して BPEL Process Manager に統合されます。実行時に、アダプタ・フレームワークによって、設計時に構成されたアダプタ・メタデータ（WSDL または J2CA 拡張要素）に基づいて、BPEL Process Manager のサービス・リクエストが J2CA コールに変換され、結果が戻されます。

設計時に生成された WSDL ファイルは、実行時に、統合するコンポーネントによって処理されます。たとえば、BPEL Process Manager はアダプタ・フレームワークを使用して、リクエスト / レスポンス・サービス（J2CA アウトバウンド相互作用）を BPEL の invoke アクティビティに統合し、アダプタ・イベントを BPEL の receive アクティビティに発行します。BPEL PM との統合の詳細は、5.2 項「アダプタと BPEL Process Manager の統合」を参照してください。

サーブレット、EJB、Java アプリケーションなどの OC4J アプリケーション・クライアントは、CCI API を使用して J2CA アダプタとインタフェースします。OC4J との統合の詳細は、5.1 項「アダプタと OC4J の統合」を参照してください。

3.4 デプロイ

パッケージ・アプリケーション・アダプタは、インストール時に、OC4J J2CA コンテナ内に J2CA 1.5 リソース・アダプタとしてデプロイされます。アダプタは、統合対象の BPEL Process Manager と同じ OC4J コンテナに存在する必要があります。

任意の Web サービス・クライアントを BSE サーブレットと統合できます。BSE は基礎となるバックエンド機能を Web サービスとして公開します。この機能は、WSDL または SOAP のいずれかです。BPEL Process Manager は、BSE レイヤーとも WSDL および SOAP バインドを介して統合可能です。

BSE は、インストール時に OC4J コンテナ内にサーブレットとしてデプロイされます。また、リモートに配置でき、BPEL Process Manager と同じコンテナ内に存在する必要はありません。

4

レガシー・アダプタ

Oracle Application Server は、レガシー・アダプタを使用して、レガシー・アプリケーションとメインフレーム・アプリケーションに統合できます。これらのアダプタには、OracleAS Adapter for Tuxedo、OracleAS Adapter for CICS、OracleAS Adapter for VSAM、OracleAS Adapter for IMS/TM および OracleAS Adapter for IMS/DB があります。

この章の内容は、次のとおりです。

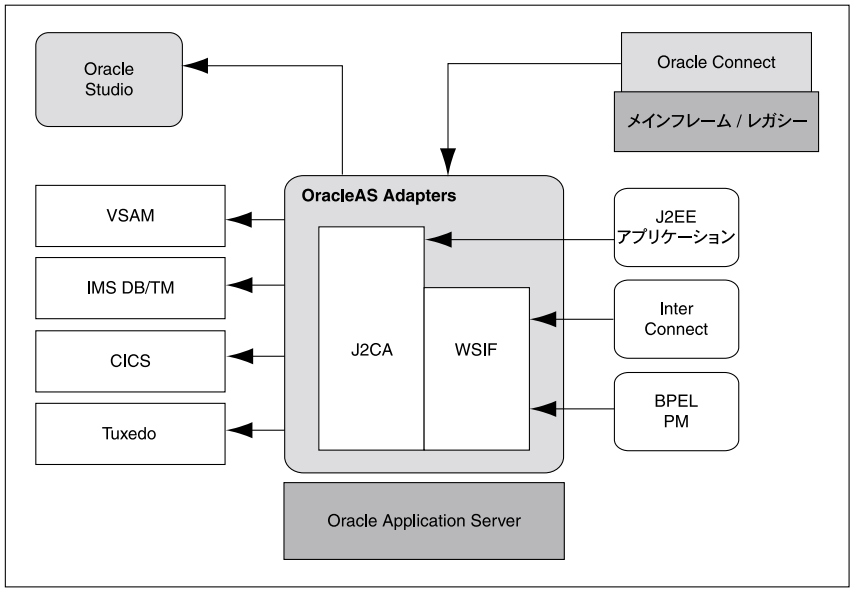
4.1 アーキテクチャ

レガシー・アダプタのアーキテクチャには、次のコンポーネントが含まれています。

- Oracle Connect
- Oracle Studio
- J2CA アダプタ

図 4-1 は、レガシー・アダプタのアーキテクチャを図で説明しています。

図 4-1 レガシー・アダプタのアーキテクチャ



4.1.1 Oracle Connect

Oracle Connect は、レガシーとメインフレーム・プラットフォームに常駐するコンポーネントです。このコンポーネントは、メインフレーム・アプリケーションおよびデータ・ストアと通信するための複数のネイティブ・アダプタで構成されています。Oracle Connect は、次のコンポーネントで構成されています。

- サーバー・プロセス
- ネイティブ・アダプタ
- デーモン
- リポジトリ

サーバー・プロセス

Oracle Connect は、クライアント・リクエストを処理する複数のサーバーで構成されています。

ネイティブ・アダプタ

Oracle Connect は、Tuxedo や IMS-TM トランザクション・システムと通信するための各種埋込みネイティブ・アダプタ、および VSAM や IMS-DB などのメインフレーム・システム上で各種データベースやファイル・システムと通信するためのデータベース・ドライバで構成されています。ネイティブ・アダプタは、レガシー COBOL アプリケーション・データなどのアプリケーション構造と XML の間で変換を実行します。メインフレーム・データと標準 XML データ間の正確なマッピングを行うために、XSD スキーマが使用されます。

デーモン

デーモンは RPC ベースのリスナーで、複数のサーバー構成を管理および維持します。デーモンは Oracle Connect が稼働するすべてのマシン上で実行され、ユーザーの認証と認可、接続割当ておよびサーバー・プロセス管理を処理します。

クライアントが接続をリクエストすると、デーモンによって、その接続を処理するためのサーバー・プロセスが割り当てられます。割り当てられるサーバー・プロセスは、新規のプロセス、またはすでに実行中のプロセスである場合があります。割当て後、クライアント・セッションとサーバー・プロセスは直接通信し、デーモンは関与しません。ただし、接続が終了してサーバー・プロセスが停止した場合、または別のクライアントで使用されている場合は、デーモンに通知されます。

デーモンは、ワークスペースと呼ばれる複数のサーバー構成をサポートしています。各ワークスペースには、アクセス可能なデータソース、アプリケーション、環境設定、セキュリティ要件およびサーバー割当てルールが定義されています。デーモンは、クライアントを認証し、特定サーバー・ワークスペース内のサーバー・プロセスに対するリクエストを認可して、クライアントに必要なサーバーを提供します。デーモンは、クライアントが使用するワークスペースに基づいてサーバーを割り当てます。したがって、クライアントは、サーバーの既存のプールからサーバー・プロセスが割り当てられた 1 つのワークスペースを使用するか、または、クライアント・リクエストごとに新規のサーバー・プロセスが割り当てられた異なるワークスペースを使用して、データソースにアクセスできます。フェイルセーフ・メカニズムによって、スタンバイとして機能する代替デーモンを指定できるため、可用性が向上します。

リポジトリ

Oracle Connect は、XML ベース・スキーマおよび構成情報を格納するためのリポジトリをサポートしています。Oracle Connect インスタンスごとに 1 つのリポジトリがあります。このリポジトリには、次の情報が格納されます。

- Oracle Connect の構成設定（クライアント / サーバー通信を制御するためのデーモン設定を含む）
- 複数のバックエンド・アプリケーションとデータソースへのシングル・サインオンを可能にするためのユーザー・プロファイル
- 各アダプタのアダプタ・メタデータ（アダプタのリクエスト / レスポンス・サービスとイベント・サービスを含む）

4.1.2 Oracle Studio

Oracle Studio は、メインフレーム用に Oracle AS Adapters を構成するための設計時ツールです。このツールを使用すると、サービス、イベント、およびネイティブ・アダプタの接続情報を構成できます。構成情報は、レガシー・アプリケーションまたはメインフレーム・アプリケーションの Oracle Connect リポジトリに格納されます。さらに、このツールを使用すると、Oracle Connect を管理および監視できます。Oracle Studio は Windows プラットフォームでのみ使用できます。Oracle Studio は IBM Eclipse GUI フレームワークに基づいています。

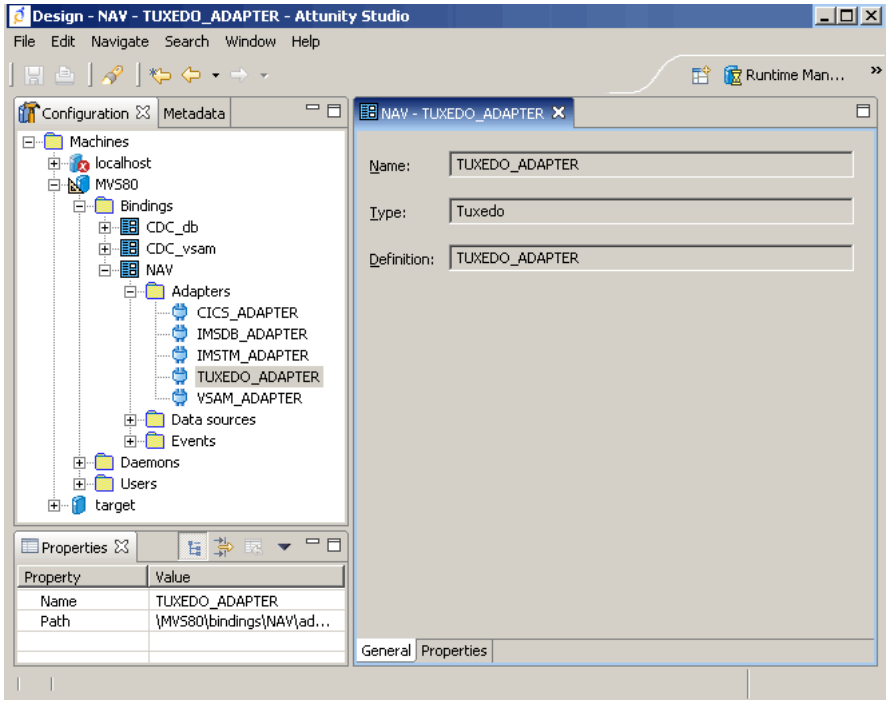
4.1.3 J2CA アダプタ

J2EE Connector Architecture（J2CA）アダプタは、OC4J アプリケーションのクライアント・リクエストを Oracle Connect アプリケーションに転送します。Oracle Connect は、メインフレーム・アプリケーションと通信して、レスポンスを J2CA アダプタに戻します。このレスポンスには、トランザクション・データや例外データ（リクエストでエラーが生成された場合）が含まれる場合があります。Business Process Execution Language for Web Services（BPEL）Process Manager および Oracle Application Server のコンポーネントは、J2CA アダプタを介して Oracle Connect と統合されています。

4.2 設計時コンポーネント

設計時にレガシー・アダプタを構成するには、次の図に示すように、Oracle Studio を使用します。

図 4-2 Oracle Studio



例 4-1 OracleAS Adapter for Tuxedo の構成

Oracle Studio を使用すると、図 4-3 に示すように、OracleAS Adapter for Tuxedo を構成できます。

図 4-3 OracleAS Adapter for Tuxedo の構成

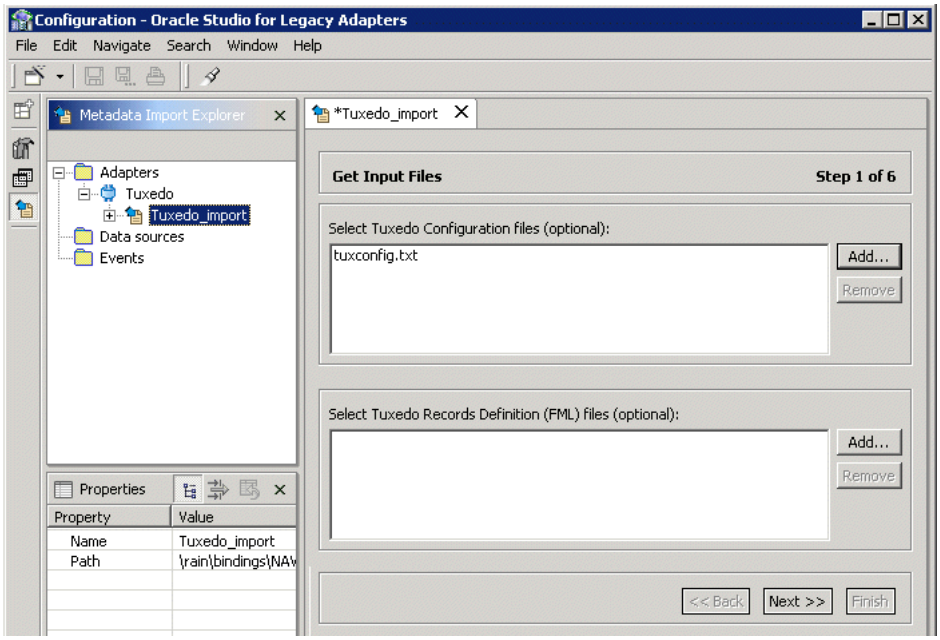
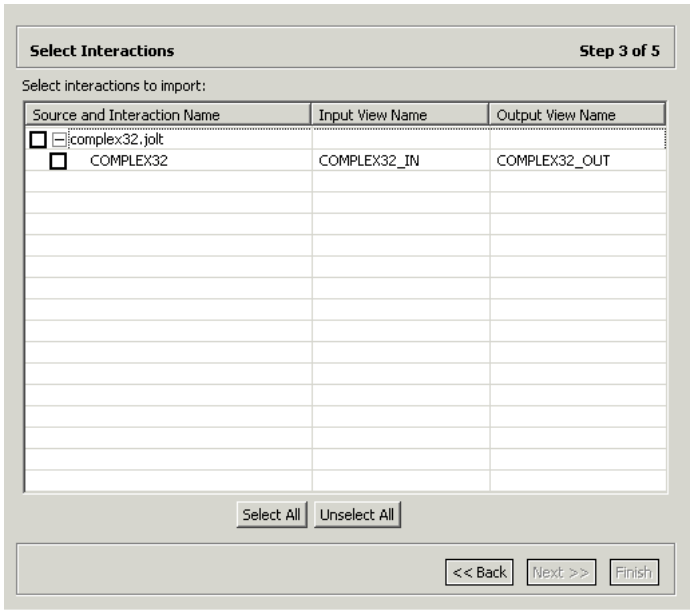


図 4-4 OracleAS Adapter for Tuxedo の相互作用タイプの選択



4.3 ランタイム・コンポーネント

設計時に生成された WSDL ファイルは、実行時に、統合するコンポーネントによって処理されます。たとえば、BPEL Process Manager はアダプタ・フレームワークを使用して、リクエスト / レスポンス・サービス（J2CA アウトバウンド相互作用）を BPEL の invoke アクティビティに統合し、イベントを BPEL の receive アクティビティに発行します。詳細は、5.2 項「アダプタと BPEL Process Manager の統合」を参照してください。

サーブレットや EJB などの OC4J アプリケーション・クライアントは、Common Client Interface（CCI）API を使用して J2CA アダプタと通信します。詳細は、5.1 項「アダプタと OC4J の統合」を参照してください。

4.4 デプロイ

レガシー・アダプタは、インストール時に、OC4J J2CA コンテナ内に J2CA リソース・アダプタとしてデプロイされます。このアダプタは、統合対象の BPEL Process Manager と同じ OC4J コンテナ内に存在する必要があります。

5

アダプタと Oracle Application Server コンポーネントとの統合

Oracle Application Server アダプタは、Oracle Containers for J2EE（OC4J）、Oracle Application Server InterConnect、Business Process Execution Language for Web services（BPEL）Process Manager、および Oracle Enterprise Service Bus など、様々なコンポーネントと統合できます。この章では、OC4J、BPEL Process Manager および OracleAS Integration InterConnect とアダプタを統合する方法を説明します。

この章の内容は、次のとおりです。

5.1 アダプタと OC4J の統合

Oracle Application Server アダプタは、OC4J コンテナ内に J2EE Connector Architecture（J2CA）1.0 リソース・アダプタとしてデプロイされます。このリソース・アダプタは、Oracle Application Server のアドレス空間で使用されます。この項では、OC4J の概要、および設計時と実行時のアダプタとの統合について説明します。この項の内容は、次のとおりです。

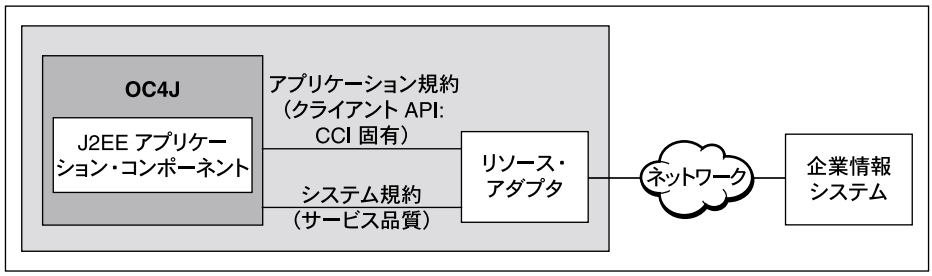
5.1.1 OC4J の概要

OC4J は Oracle Application Server のコア J2EE ランタイム・コンポーネントです。OC4J は標準 JDK 1.4 Java 仮想マシン（JVM）で動作する J2EE 1.3 準拠のコンテナであり、JavaServer Pages（JSP）ファイル、サーブレット、Enterprise JavaBeans（EJB）、Web サービスおよびすべての J2EE サービスを完全にサポートしています。さらに、OC4J は、J2CA リソース・アダプタをホスティングするための J2CA コンテナで構成されています。J2CA は、J2EE サーバーと各種バックエンド・アプリケーションの統合を簡略化するための標準 Java インタフェースを定義します。

すべてのクライアント・アプリケーションは、OC4J 環境内で動作します。OC4J クライアント・アプリケーションをリソース・アダプタと統合するには、Common Client Interface（CCI）を使用します。OC4J アダプタ・クライアントには、CCI Application Program Interface（API）を実装するサーブレット、EJB または Java アプリケーション・クライアントが含まれます。CCI は、アプリケーション・コンポーネントがバックエンド・アプリケーションにアクセスするための標準クライアント API を定義します。

これに対して、OC4J コンテナとリソース・アダプタ間の規約は、Service Provider Interface（SPI）によって定義されます。この規約によって、OC4J とアダプタ間の標準が定義されます。これらの規約はシステムによって自動的に処理され、アプリケーション開発者には明示されません。次の図は、CCI と SPI 間の規約を説明しています。

図 5-1 OC4J とリソース・アダプタ間の規約



OC4J アーキテクチャには、次のシステム・レベルの規約があります。

- 接続管理 : この規約によって、アプリケーション・コンポーネントは、バックエンド・アプリケーションに接続し、OC4J コンテナの接続プーリング・サポートを活用できます。接続管理によって、バックエンド・アプリケーションへのアクセスが必要な多数のコンポーネントをサポートできる、スケーラブルで効率的な環境が実現します。
- トランザクション管理 : この規約によって、アプリケーション・サーバーは、トランザクション・マネージャを使用して複数のリソース・マネージャにわたってトランザクションを管理できます。ほとんどのアダプタは、ローカル・トランザクション（1 フェーズのコミット）のみをサポートし、XA トランザクション（2 フェーズのコミット）はサポートしていません。
- セキュリティ管理 : この規約によって、J2EE サーバーとバックエンド・アプリケーション間の、認証、認可およびセキュアな通信が提供されます。OC4J は、コンテナ管理のサインオン（Oracle Application Server によりセキュリティ資格証明がバックエンド・アプリケーションに送信されます）およびコンポーネント管理のサインオン（J2CA アダプタによりセキュリティ資格証明がバックエンド・アプリケーションに転送されます）の両方をサポートします。

5.1.2 OC4J とアダプタの統合

Oracle Application Server アダプタは J2CA 1.5 ベースのアダプタで、インストール時に OC4J 内にデプロイされます。このリリースでの OC4J コンポーネントがサポートするのは、J2CA 1.0 仕様のみです。J2CA リソース・アダプタは、Java アーカイブ（JAR）形式を使用して、リソース・アダプタ・アーカイブ（RAR）ファイルにパッケージ化されています。RAR ファイルには、正しく書式設定されたデプロイメント・ディスクリプタ（`/META-INF/ra.xml`）が含まれています。さらに、OC4J とリソース・アダプタ間の規約に関する宣言情報も含まれています。

OC4J では、J2CA アダプタのデプロイ時に、対応する `oc4j-ra.xml` ファイルが生成されます。この `oc4j-ra.xml` ファイルは、リソース・アダプタ用のデプロイメント・ディスクリプタです。このファイルには、リソース・アダプタを OC4J にデプロイするためのデプロイ構成が含まれており、これには、リソース・アダプタのデプロイメント・ディスクリプタに指定されているバックエンド・アプリケーションの接続情報、使用する Java Naming and Directory Interface（JNDI）の名前、接続プーリング・パラメータ、およびリソース・プリンシパル・マッピング・メカニズムと構成が含まれます。

設計時

アダプタのリクエスト / レスポンス・サービス用に XML スキーマ定義（XSD）ファイルを生成するには、アダプタ設計時ツールを使用します。OC4J クライアントは、実行時にこの XSD ファイルを使用して、J2CA アウトバウンド相互作用をコールします。パッケージ・アプリケーション・アダプタは OracleAS Adapter Application Explorer（Application Explorer）を、レガシー・アダプタは OracleAS Studio を、テクノロジ・アダプタは JDeveloper をそれぞれ使用します。

実行時

Oracle Application Server アダプタは J2CA 1.5 仕様に基づいていますが、このリリースでは、J2CA 1.0 リソース・アダプタとして OC4J コンテナ内にデプロイされています。J2CA 1.0 仕様は、リクエスト / レスポンス・サービス（アウトバウンド相互作用とも呼ばれます）に対応していますが、アダプタによるバックエンド・イベントの非同期発行には対応していません。J2CA 1.5 仕様は、ライフ・サイクル管理、メッセージ・インフロー（アダプタ・イベント発行用）および作業管理の規約に対応しています。

5.2 アダプタと BPEL Process Manager の統合

Oracle Application Server アダプタは BPEL Process Manager に統合できます。この項の内容は、次のとおりです。

5.2.1 BPEL Process Manager の概要

BPEL Process Manager は、BPEL ビジネス・プロセスを作成、デプロイおよび管理する包括的なソリューションです。BPEL Process Manager はサービス指向アーキテクチャ（SOA）に基づいて、柔軟性、相互運用性、再利用性、拡張性および迅速な実装を提供します。BPEL Process Manager を使用すると、既存のビジネス・プロセスの管理、変更、拡張および再デプロイ全般にわたるコストを削減できます。各ビジネス・アクティビティは自己完結型で自己記述的なモジュール・アプリケーションであり、WSDL ファイルで定義されたインタフェース、および Web サービスとしてモデル化されたビジネス・プロセスを使用します。

次の項では、テクノロジ・アダプタが、BPEL Process Manager 10.1.3 内で、インバウンドまたはアウトバウンドで処理するすべてのメッセージの統計を、どのように収集および公開するかを説明します。

5.2.1.1 アダプタ統計

BPEL Process Manager 10.1.3 では、ファイル、JMS、データベースなどのテクノロジ・アダプタが、インバウンドまたはアウトバウンドで処理するすべてのメッセージについて、統計を収集および公開します。この統計はカテゴリおよび各タスクに分類されています。アウトバウンド・プロセスにおける統計の分類の例を次に示します。

- アダプタ前処理
  - `InteractionSpec` の準備
- アダプタ処理
  - コール可能文の設定
  - データベースの起動
  - 結果の解析
- アダプタ後処理

このアダプタ統計は、BPEL コンソールから、BPEL エンジンの統計と同じページで参照できます。次に、BPEL コンソールでアダプタ統計を表示する手順を説明します。

1. BPEL PM コンソールのトップページのリンク「BPEL ドメインの管理」を進みます。
2. 次ページで、リンク「アダプタ統計」をクリックします。

   ここに現在アクティブなインバウンドおよびアウトバウンドのアダプタ相互作用の全リスト、および各アダプタが実行する様々なステップの平均実行時間が示されています。

   統計収集をリセットするには、リンク「統計データの消去」をクリックします。

5.2.2 BPEL Process Manager とアダプタの統合

J2CA 1.5 リソース・アダプタと BPEL Process Manager の双方向統合には、アダプタ・フレームワークが使用されます。アダプタ・フレームワークは標準に基づいており、Web Service Invocation Framework（WSIF）テクノロジを採用して、基礎となる J2CA 相互作用を Web サービスとして公開します。

設計時

アダプタを BPEL Process Manager に統合する間、基礎となるアダプタ・サービスは、J2CA 拡張要素が指定された WSDL ファイルとして公開されます。次の表は、各種アダプタ用に WSDL ファイルを生成する際に使用される設計時ツールのリストです。

アダプタ	ツール
テクノロジおよび OracleAS Adapter for Oracle Applications	JDeveloper
パッケージ・アプリケーション	Application Explorer
レガシー	Oracle Studio

WSDL ファイルは、アダプタのリクエスト / レスポンス・サービスとイベント通知サービスの両方に対して作成されます。J2CA 拡張機能には J2CA 要素が含まれています。これらの要素は、実行時に J2CA 相互作用との間で Web サービス・メッセージの変換を行うアダプタ・フレームワークに必要です。J2CA WSDL 拡張要素には、アダプタ・フレームワークがリクエスト / レスポンス・サービスをコールし、インバウンドの J2CA 1.5 エンドポイントをアクティブ化してインバウンド・イベントを受信するためのメタデータが含まれます。リクエスト / レスポンス・サービスの J2CA 拡張要素には、アウトバウンド相互作用をコールするための JNDI ロケーションおよび `InteractionSpec` 詳細が含まれます。イベント通知サービスの J2CA 拡張要素には、J2CA インバウンド相互作用を介してアダプタ・イベントを発行するためのリソース・アダプタ・クラス名および `ActivationSpec` パラメータが含まれます。

図 5-2 は、テクノロジ・アダプタで使用する設計時ツールの JDeveloper を図で説明しています。

図 5-2 テクノロジ・アダプタの設計時の構成

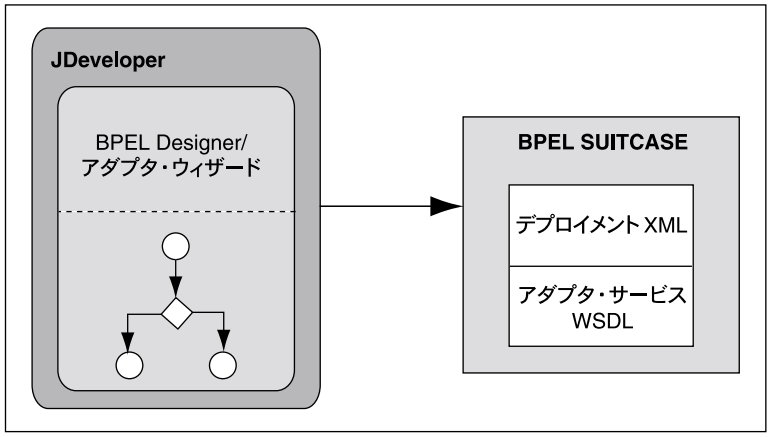
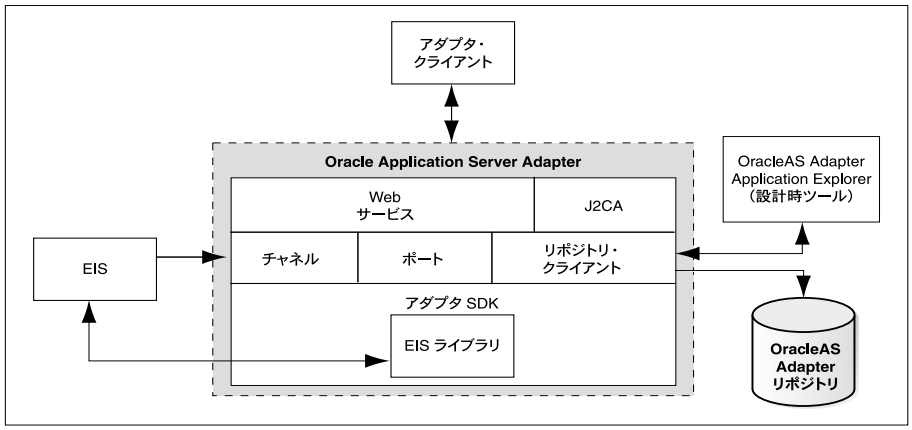


図 5-3 は、パッケージ・アプリケーション・アダプタとレガシー・アダプタを構成するための設計時ツールを図で説明しています。この図では、アダプタ・メタデータを WSDL ファイルとして公開するために、設計時ツールが使用されています。この WSDL ファイルは、実行時に BPEL Process Manager によって処理されます。

図 5-3 レガシー・アダプタとパッケージ・アプリケーション・アダプタの構成



実行時

Oracle Application Server アダプタは J2CA 1.5 仕様に基づいており、Oracle BPEL Process Manager を使用して同じ OC4J コンテナにデプロイされます。アダプタ・フレームワークは、実行時に標準 J2CA 1.5 リソース・アダプタを Oracle BPEL Process Manager と統合するグルー・レイヤーとして機能します。Oracle Application Server OC4J コンテナのサポート対象は J2CA 1.0 のみですが、アダプタ・フレームワークは疑似 J2CA 1.5 コンテナとして機能し、実装を可能にします。

アダプタ・フレームワークは、J2CA 相互作用を Web サービスとしてラップするための WSIF J2CA プロバイダで構成されています。さらに、アダプタ・フレームワークは、設計時に生成された WSDL ファイルに基づいて、Web サービス・メッセージを J2CA 相互作用メッセージに変換します。

WSIF プロバイダは、Web サービス起動（BPEL PM の invoke アクティビティからの）を J2CA アウトバウンド相互作用コールに変換し、逆方向の変換も実行します。つまり、BPEL の invoke アクティビティによって起動した Web サービス起動は、J2CA CCI アウトバウンド相互作用に変換され、J2CA レスポンスは元の Web サービス・レスポンスに変換されます。このエ

ンドツーエンドの起動は同時に行われます。WSIF プロバイダは、一方向の非同期 J2CA アウトバウンド相互作用の起動もサポートしています。WSIF プロバイダ要素では、J2CA 実装詳細を BPEL Process Manager プロセスに明示せず、Web サービス詳細を J2CA 1.5 リソース・アダプタに明示しません。

BPEL Process Manager とアウトバウンド相互作用の統合

BPEL Process Manager は、WSIF テクノロジを使用して、リソース・アダプタのリクエスト / レスポンス・サービスを開始します。WSIF は、Web サービスをコールし、Web サービス定義の抽象部分を物理バインド（転送とプロトコル）から分離して、サービス指向アーキテクチャ（SOA）を生成するために使用されます。次に、BPEL Process Manager とアウトバウンド相互作用を統合するプロセスを要約して説明します。

- アダプタ・サービスが設計時に WSDL ファイルとして公開され、BPEL プロセスの `PartnerLink` アクティビティの構成時に処理されます。
- WSDL 拡張機能によって、リソース・アダプタの JNDI アドレス、`InteractionSpec` クラス名および `InteractionSpec` パラメータが指定されます。
- 実行時に、BPEL Process Manager の invoke アクティビティを使用して、J2CA リソース・アダプタのアウトバウンド相互作用である `PartnerLink` アクティビティがコールされます。
- WSIF J2CA プロバイダが BPEL Process Manager から Web サービス・リクエストを受信します。
- この Web サービス・リクエストに関連付けられている WSDL ファイルには、WSDL 拡張要素内の J2CA `ConnectionFactory` クラスの JNDI 名が格納されています。
- WSIF プロバイダが JNDI 参照を実行して Connection インスタンスを取得します。
- WSIF プロバイダが WSDL 拡張要素 jca:operation から `InteractionSpec` Bean オブジェクトを作成します。
- WSIF プロバイダは、`InteractionSpec`、`InputRecord` および空の `OutputRecord` パラメータを渡して J2CA アウトバウンド相互作用を開始します。
- J2CA 1.5 リソース・アダプタがバックエンド・アプリケーションをコールし、バックエンド・アプリケーション・レスポンスを WSIF プロバイダに戻します。
- この XML レコードが Web サービス・レスポンスに変換され、BPEL PM インスタンスによって処理されます。
- このエンドツーエンドのシナリオ全体は同時に実行され、J2CA 例外が発生した場合は Web サービス障害メッセージとして転送されます。

BPEL Process Manager とインバウンド相互作用の統合

BPEL Process Manager は、疑似 J2CA 1.5 コンテナであるアダプタ・フレームワークを介して J2CA 1.5 リソース・アダプタからイベントを受信し、アダプタからイベントを受信するためのメッセージ・インフロー規約を実装します。J2CA インバウンド相互作用は、設計時に WSDL ファイルに取得されます。J2CA インバウンド WSDL バインド・セクションには、J2CA 1.5 `ActivationSpec` パラメータが含まれます。`ActivationSpec` パラメータによって、インバウンド接続およびインバウンド相互作用詳細が（J2CA 1.5 仕様に従って）取得されます。J2CA インバウンド WSDL サービス・セクションには、J2CA 1.5 `ResourceAdapter` クラス名が含まれます。さらに、サービス・セクションには、オプションで JNDI ロケーションを設定できます。次に、BPEL Process Manager とインバウンド相互作用を統合するプロセスを要約して説明します。

- `ResourceAdapter` クラス名と `ActivationSpec` パラメータが、設計時に J2CA インバウンド相互作用 WSDL の WSDL 拡張セクションに取得され、実行時に BPEL Process Manager とアダプタ・フレームワークで使用可能になります。
- J2CA 1.5 `ResourceAdapter` クラスのインスタンスが作成され、その J2CA `ResourceAdapter` クラスの `Start` メソッドがコールされます。

- BPEL Process Manager プロセスでインバウンド相互作用操作が参照されるたびに、J2CA 1.5 ResourceAdapter インスタンスの EndPointActivation メソッドが起動します。アダプタ・フレームワークは、J2CA インバウンド相互作用の WSDL 拡張セクションに存在する ActivationSpec 詳細に基づいて ActivationSpec クラス（JavaBeans）を作成し、J2CA 1.5 リソース・アダプタのエンドポイントをアクティブ化します。
- アダプタ・フレームワークの MessageEndpoint 実装では javax.resource.cci.MessageListener インタフェースが実装されます。J2CA 1.5 リソース・アダプタは、バックエンド・アプリケーション・イベントを受信した際に、この MessageEndpoint で onMessage() メソッドをコールします。J2CA 1.5 リソース・アダプタは、endpointActivation の処理中にリソース・アダプタに提供された MessageEndpointFactory を介して、MessageEndpoint 実装のインスタンスを作成します。
- アダプタ・フレームワークが MessageListener クラスを介してイベントを受信し、そのイベントを BPEL Process Manager インスタンスの receive アクティビティに転送します。
- BPEL プロセスが停止した場合、関連するすべてのインバウンド・エンドポイントは、リソース・アダプタにより実装されている endPointDeactivation メソッドを介して非アクティブ化されます。

BPEL WSIF JCA 接続プール

J2CA アダプタの場合、特に DB アダプタや AQ アダプタなど JDBC ベースのものついては、2 種類の接続管理が行われます。1 つはインバウンド（エンドポイント）のアクティブ化（BPEL 受信）用、もう 1 つはアウトバウンド相互作用（BPEL 起動）用です。

インバウンドのアクティブ化の場合、J2CA アダプタが接続作成およびリカバリ全般を管理します。アダプタ・フレームワーク（AF）は、J2CA ConnectionFactory ハンドルをルックアップし、ActivationSpec を介してアダプタに提供するためにのみリクエストされます。これは、接続の作成に使用可能な、Application Server 接続マネージャを経由する特定のインタフェースを実装している場合にのみ可能です。管理対象（JDBC）接続がうまくいかない場合、アダプタは、J2CA 接続ハンドルを閉じ（Application Server で destroy() がコールされた場合は、これに続いて管理対象接続も閉じ)、テンポラリ・リカバリ・ループに入り、新しい接続の確立を試みます。

アウトバウンド相互作用（WSIF または J2CA）の場合には、各 WSIF ポートは次のタプルをキャッシュします。

- ConnectionFactory
- ConnectionSpec
- Connection
- Interaction
- InteractionSpec

BPEL エンジンでは通常、WSIF ポートがいくつかの数のスレッドと同時に起動されるため、キャッシュ・サイズには常にその時点で最も高い並行処理レベルが反映されています。このキャッシュは、設定したアイドル期間後に使用されていないタプルを自動的に期限切れにするように調整できます（その後、相互作用および接続ハンドルもクローズします)。このキャッシュによって、高負荷の環境、たとえば Retek（800 万トランザクション / 時）などの環境におけるパフォーマンスを大幅に改善できます。

キャッシュを使用する JCA アダプタ相互作用の 1 つのみが ResourceException をスローしても、キャッシュのすべてのメンバーがクローズされ即時解放（パージ）されるので、新しい相互作用はキャッシュに（新規に）メンバーを再作成する必要があります。BPEL エンジンには、パートナ・リンク再試行という機能があります。この機能は、起動のたびに実行されるように設定できます。これにより、JCA アダプタの起動または相互作用で ResourceException が Retryable とマークされてスローされた場合はすべて、エンジンで起動（DB 更新）が再試行され、この起動によって WSIF ポート・キャッシュが再移入されます（DB が再び使用可能となった場合。主に RAC の場合で即時)。

ファイル・アダプタなどの非トランザクション・アダプタ（adapterMetadata.supportsLocalTransactionDemarcation() == false）の場合は、J2CA 接続キャッシュには 1 つのメンバーしか含まれていません。そのため、入ってくるスレッドはすべて同じ CCI 接続ハンドルに多重に送られます。

JCA 接続キャッシュは、次の bpel.xml のパートナ・リンク・プロパティを使用して明示的に有効化または設定できます。

```
<property name="useJCAConnectionPool">true</property>
```

通常、このプロパティはアダプタの宣言済のトランザクション・サポートから導出されます。たとえば、ファイル・アダプタではこの接続プールをマルチスレッド・セーフであるため使用しませんが、これは次のプロパティで上書きできます。

```
<property name="maxSizeJCAConnectionPool">500</property>
```

前述の例のプロパティが指定されていない場合、接続プールのサイズは無制限とみなされます。これは、WSIFPort（パートナ・リンク）1 つ当たりを基準に適用されます。

```
<property name="lruConnectionMaxIdleAge">50000</property>
```

プール内のアイドル接続の最長期間は、接続のタイプによっては高コストの外部リソースを保持することになるため、重要です。たとえば、ミリ秒で計測される DB のシャドウ・プロセスなどです。次に例を示します。

```
property name="lruConnectionCheckInterval">10000</property>
```

前述の例のプロパティでは、アイドル接続についてどのくらいの頻度で接続プールをスキャンするかを、これもミリ秒単位で決定しています。

5.3 アダプタと OracleAS Integration InterConnect の統合

OracleAS Adapter for PeopleSoft、OracleAS Adapter for SAP R/3、OracleAS Adapter for Siebel および OracleAS Adapter for J.D. Edwards を含むパッケージ・アプリケーション・アダプタは、OracleAS Integration InterConnect に統合できます。J2CA デプロイ以外に、パッケージ・アプリケーション・アダプタは、Business Service Engine（BSE）と呼ばれる Web サービス・サーブレット・デプロイをサポートしています。この統合は、エンタープライズ内、またはインターネットを介してエンタープライズ境界間でデプロイできます。OracleAS Integration InterConnect は、ハブ・アンド・スポーク・システムとして設計されています。このシステムでは、スポークは、他のアプリケーションおよびシステムにアクセスする OracleAS Integration InterConnect アダプタとして機能します。ハブは、スタンドアロン Java アプリケーションである OracleAS Integration InterConnect リポジトリ・サーバーとして機能します。OracleAS Integration InterConnect リポジトリは、設計時のメタデータ定義を保存するデータベースです。

BSE を介したデプロイとの統合

BSE は OracleAS Integration InterConnect 製品と統合され、パッケージ・アプリケーションにアクセスできます。インターネットによって異機種間情報システムに単純で一貫性のある接続を提供する標準を実装することで、このようなシステムをより迅速かつ高いコスト効率で一様に統合できます。ビジネス機能およびデータは Web Service として公開され、その他のプラットフォーム上のツールでは、これらのサービスを最小限の労力で使用できます。OracleAS Adapter Business Services Engine は、次の標準をサポートしています。

- Simple Object Access Protocol（SOAP）リクエストにより、Web Service でタスクが実行されます。
- Web Services Definition Language（WSDL）は、一連のサービスに使用できるすべての SOAP コールを記述します。
- Universal Description, Discovery, and Integration（UDDI）サーバーは、関連する Web Service のディレクトリ、および Web Service を記述する WSDL として機能します。

これらのサービスを生成するためのプログラミングの知識は不要です。これらのサービスは、ERP または CRM システム、レガシー・アプリケーションおよび様々なデータ・ソースの機能を再利用します。

BSE は Oracle Application Server の OC4J コンテナ内のサーブレットとしてデプロイされ、スケーラビリティおよび高可用性については OC4J コンテナに依存します。Swing ベースの設計時ツールである Application Explorer は、BSE に渡される EIS メタデータを参照するために使用されます。アダプタは、様々なアダプタ・サービス用の WSDL を作成し、XML ベースのファイル・リポジトリまたは Oracle Database リポジトリに WSDL スキーマを格納します。BSE は、設計時および実行時に実行する必要があります。

OracleAS Integration InterConnect との統合

OracleAS Integration InterConnect EIS Adapter Plugin は、XSD 形式と DTD 形式の間で XML ペイロードを変換することで、OracleAS Web Services アダプタを OracleAS Integration InterConnect と統合します。

OracleAS Integration InterConnect 製品は、実装済プロシージャ（リクエスト / レスポンス・サービス）、サブスクライブ（一方向のリクエスト・サービス）、パブリッシュ（イベント・サービス）および起動済プロシージャの 4 つのメッセージ・パラダイムをサポートしています（この場合、OracleAS Integration InterConnect はサーバーであり、EIS はリクエストを行うクライアントです）。

OracleAS Integration InterConnect EIS Adapter Plugin は、最初の 3 つのメッセージ・パラダイムのみをサポートしています。実装済プロシージャおよびサブスクリプションには SOAP、リクエスト / レスポンスおよびイベント通知サービスには RMI が使用されます。

OracleAS Integration InterConnect との統合には、次のコンポーネントが使用されます。

- BSE（OC4J 内の Web Service サーブレットとしてデプロイされる OracleAS Adapter）
- OracleAS Integration InterConnect EIS Adapter Plugin（BSE との対話に使用）
- OracleAS Integration InterConnect デプロイ
- Oracle iStudio
- Application Explorer

設計時の OracleAS Integration InterConnect

Application Explorer は設計時に BSE を構成します。標準の WSDL スキーマ以外に、BSE は DTD スキーマを生成し、ユーザーが指定したファイル・システム内にローカルに保存します。OracleAS Integration InterConnect iStudio ツールには DTD ブラウザ・プラグインが含まれています。この DTD ブラウザ・プラグインを使用すると、DTD スキーマをネイティブの OracleAS Integration InterConnect スキーマに変換し、OracleAS Integration InterConnect リポジトリに格納できます。具体的には、設計時に次の手順でリクエスト / レスポンスおよびイベント通知のシナリオを構成します。

リクエスト / レスポンス : 実装済プロシージャおよびサブスクライブ

1. Application Explorer を使用して、リクエスト / レスポンス・サービスの BSE を構成します。WSDL スキーマは OracleAS Adapter リポジトリに保存され、DTD スキーマはローカル・ファイル・システムに保存されます。
2. OracleAS Integration InterConnect iStudio DTD ブラウザを使用して、前の手順で生成した DTD スキーマを処理し、構成情報を OracleAS Integration InterConnect リポジトリに保存します。

イベント通知 : パブリッシュ

1. Application Explorer を使用して、RMI ポート定義を含むイベント・サービスの BSE を構成します。WSDL スキーマは Oracle Adapter リポジトリに保存され、DTD スキーマはローカル・ファイル・システムに保存されます。
2. OracleAS Integration InterConnect iStudio DTD ブラウザを使用して、前の手順で生成した DTD スキーマを処理し、構成情報を OracleAS Integration InterConnect リポジトリに保存します。

実行時の OracleAS Integration InterConnect

OracleAS Integration InterConnect EIS Adapter Plugin は、独自の API を使用して、OracleAS Integration InterConnect リポジトリからメタデータをロードします。OracleAS Integration InterConnect EIS Adapter Plugin は EIS アプリケーションとインタフェースして、EIS ネイティブ形式と「アプリケーション・ビュー」(OracleAS Integration InterConnect で認識される形式)の間でデータを変換します。この相互作用を図 6 に示します。次の手順で、より具体的にリクエスト / レスポンスおよびイベント通知のシナリオの実行時の動作を示します。

リクエスト / レスポンス : 実装済プロシージャおよびサブスクライブ

OracleAS Integration InterConnect ハブは、OracleAS Integration InterConnect EIS Adapter Plugin にリクエストを送信します。OracleAS Integration InterConnect EIS Adapter Plugin は、SOAP クライアント・リクエストを作成して、BSE をコールします。アダプタは EIS を起動して、EIS レスポンスを SOAP レスポンスとして OracleAS Integration InterConnect EIS Adapter Plugin に転送します。OracleAS Integration InterConnect EIS Adapter Plugin は、SOAP レスポンスを適切な OracleAS Integration InterConnect 形式に変換して、OracleAS Integration InterConnect ハブに転送します。

イベント通知 : パブリッシュ

BSE は、チャネル・コンポーネントを介して EIS イベント通知を受信します。このイベントは、RMI リクエストとして OracleAS Integration InterConnect EIS Adapter Plugin に転送されます。OracleAS Integration InterConnect EIS Adapter Plugin は RMI サーバーとして機能してこのイベントを処理し、適切な OracleAS Integration InterConnect 形式に変換して、OracleAS Integration InterConnect ハブに転送します。

J2CA を介したデプロイとの統合

OracleAS Adapter J2CA は、リクエスト / レスポンス・サービスなどの同期アウトバウンド相互作用のみをサポートします。イベント通知はサポートしません。実行時に、EJB、サーブレットまたは Java アプリケーション・クライアントは、CCI API を介して OracleAS Adapter J2CA と通信します。OracleAS Adapter J2CA は、標準の J2CA 1.0 Resource Adapter として OC4J 内にデプロイされ、OC4J コンテナ内で管理モードで実行されます。OracleAS Adapter J2CA は、BSE と同様に、Application Explorer を使用して構成されます。OracleAS Adapter J2CA は、実際にはアダプタ SDK の J2CA ラッパーです。

OracleAS Adapter J2CA は汎用的です。OracleAS Adapter J2CA のリポジトリ・プロジェクトには、複数の EIS 接続を含めることができ、複数の EIS タイプと対話できます。リポジトリ・プロジェクトは、ファイルまたはデータベース・リポジトリに格納できます。`ra.xml` ファイルには、リポジトリ接続パラメータおよびリポジトリ・プロジェクト名が含まれます。OC4J によって生成される `oc4j-ra.xml` ファイルには、複数の管理 ConnectionFactory オブジェクトを指定できます。各オブジェクトは、異なるリポジトリ・プロジェクトと JNDI 名を指します。特定の EIS に接続するには、J2CA CCI ConnectionSpec クラスを使用して EIS タイプおよび EIS 接続名を指定します。

OracleAS Adapter J2CA で CCI を使用するには、次の手順を実行します。

1. OracleAS Adapter J2CA の `ConnectionFactory` オブジェクトを検索します。
2. EIS タイプおよび EIS 接続名を使用する `ConnectionSpec` オブジェクトを作成します。
3. `ConnectionFactory` および `ConnectionSpec` オブジェクトを使用して、EIS への `Connection` を作成します。
4. リクエストおよびレスポンス XSD スキーマに基づいて、`Interaction` オブジェクトを作成します。
5. Interaction で `execute()` メソッドをコールします。
6. 必要な相互作用が処理された後、`Interaction` および `Connection` オブジェクトを閉じます。

5.4 アダプタと Oracle Enterprise Service Bus の統合

この項の内容は次のとおりです。

5.4.1 Oracle Enterprise Service Bus の概要

Enterprise Service Bus（ESB）は、エンタープライズ内外を問わず複数のエンドポイント間でデータを移動させます。Enterprise Service Bus（ESB）は、オープン・スタンダードを使用して、多様なアプリケーション間でビジネス文書を（Extensible Markup Language（XML）メッセージとして）接続、変換およびルーティングします。これにより、既存のアプリケーションにあまり影響を与えることなく、ビジネス・データの監視および管理が可能になります。Enterprise Service Bus は、サービス指向アーキテクチャ（SOA）およびイベントドリブン・アーキテクチャ（EDA）を提供するための基盤となるインフラストラクチャです。

Oracle Enterprise Service Bus は、SOA および EDA を使用するサービスのためのベースとなります。そのコアの部分は疎結合されたアプリケーション・フレームワークで、業界標準を使用する、異機種間に分散されたメッセージ指向の環境において、ビジネスに柔軟性、再利用性および全面的な応答性をもたらします。

接続性は、アダプタ・サービスおよび Simple Object Access Protocol（SOAP）呼出しサービスを通じて提供されます。

5.4.2 Oracle Enterprise Service Bus とアダプタの統合

Oracle JDeveloper ESB Designer でサービスを作成して、Oracle Enterprise Service Bus を、ファイル・システム、データベース表、データベース・キュー、Java Message Services（JMS）、および Oracle E-Business Suite とすべての SOAP サービスに統合できます。

サービスは Enterprise Service Bus の主要部分です。Oracle Enterprise Service Bus の設計では、様々なサービスを作成し、メッセージが Service Bus へ移動し、Service Bus 内を通り、Service Bus から出ていくようにします。

Oracle Enterprise Service Bus へのデータの移動

Service Bus へデータを移動させるには、インバウンド・アダプタ・サービスを使用するか、外部アプリケーションから ESB サービスがコールされるようにします。

Oracle Enterprise Service Bus からのデータの移動

Service Bus からデータを移動させるには、アウトバウンド・アダプタ・サービスを使用するか、外部 SOAP サービスを起動します。

Oracle Enterprise Service Bus 内でのデータの移動

Service Bus 内でデータを移動させ、データ構造をソース・アプリケーションでの構造からターゲット・アプリケーションで必要とされる構造へ変換させるには、ルーティング・サービスを使用します。

Oracle Enterprise Service Bus では、Oracle Technology アダプタ用のサービスの作成がサポートされています。Oracle Technology アダプタを使用することにより、メインフレームおよびレガシー・アプリケーションを Enterprise Resource Planning（ERP）、Customer Relationship Management（CRM）、データベースおよびメッセージ・システムに統合できます。

5.4.2.1 アダプタ・サービスの概要

Oracle JDeveloper ESB Designer では、インバウンドおよびアウトバンドのアダプタ・サービスの作成を補助するウィザードが提供されます。このウィザードでは、サービスを定義する WSDL ファイルを生成するために必要な情報を収集します。

アダプタ・サービスは、インバウンドまたはアウトバウンドのアダプタ・サービスとして構成できます。インバウンド・アダプタ・サービスは Oracle Enterprise Service Bus にメッセージを送信し、アウトバウンド・アダプタ・サービスは Oracle Enterprise Service Bus の外部のアプリケーションまたはシステムにメッセージを送信します。

5.4.2.2 Oracle Enterprise Service Bus でサポートされる / サポートされないアダプタ・フレームワークの機能

この項では、アダプタ・フレームワークの機能について、リリース 10.1.3 の Oracle Enterprise Service Bus でサポートされるものとサポートされないものを説明します。

Oracle Enterprise Service Bus でサポートされるアダプタ・フレームワークの機能

次の機能は、リリース 10.1.3 の Oracle Enterprise Service Bus でサポートされます。

パートナ・リンク / アクティブ化エージェント（bpel.xml）のプロパティ

リリース 10.1.3 では、アダプタ・フレームワークは、`oracle.tip.esb.jca.ESBActivationAgent.activateEvent` を介して `ESBActivationAgent` に渡されたパートナ・リンクおよび / またはアクティブ化エージェントのプロパティを、`java.util.Map` を介して受け取ります。

このマップの起点は BPEL 内で使用可能な `bpel.xml`（BPEL アクティブ化フレームワークで提供）ですが、どのようなタイプの名前または値セットもソースにすることが可能です。これらのプロパティは、最終的に `EndpointPropertiesAssociation` インタフェースを通じて `ResourceAdapter` に提供されます。

アウトバウンド WSIF での `execute()` の再試行

WSIF 操作の `executeRequestResponseOperation()` メソッドは、`oracle.tip.esb.serviceimpl.outadapter.OutboundAdapterService.nextService()` メソッド内で起動します。

リリース 10.1.3 では、すべての `WSIFException` が `EsbServiceException` にリンクされ、`processBusinessEvent` から伝播します。

`WSIFException` には `isRetryable()` というブール・フラグがあり、サービス・コールを再試行するか中断するかを決めるために使用できます。

この再試行の編成は、BPEL で現行使用されているものと同じパートナ・リンク・プロパティによって制御できます。たとえば `retryInterval`（秒）、`retryMaxCount` などです。`retryMaxCount` はオプションで、省略された場合は無制限となることに注意してください。

インバウンドでの `onMessage()` の再試行

リリース 10.1.3 では、`ESBListenerImpl.onMessage()` からはどのような例外もリソース・アダプタに戻されません。ただし、次の 2 つの JCA ResourceException フレーバを考慮する場合があります。

リカバリ不能な（再試行しない）ケース

`oracle.tip.adapter.api.exception.PCResourceException`（`ResourceException` を拡張）

リカバリ可能な（再試行する）ケース

`oracle.tip.adapter.api.exception.PCRetriableResourceException`（`PCResourceException` を拡張）

インバウンドでの拒否の処理

アクティブ化エージェント（bpel.xml）のプロパティは現在サポートされていないため、アダプタ・フレームワーク・サポートの拒否処理全般は使用可能ではありません。ただし、デフォルトの拒否ハンドラが使用できるため、明示的に宣言された例外ハンドラがない場合は、このハンドラを作動できます。

致命的なエラーの処理

ESBListenerImpl.onFatalError() が起動した場合、ESBListenerImpl.java は現在のところ、単に JCA EndpointDeactivation() を起動します。この結果、リソース・アダプタの作業スレッドが終了します（イベントのプーリングは停止します）。

アラート

ESBListenerImpl.onAlert() が起動した場合、ESBListenerImpl.java は現在のところ、特に何もしません。onAlert() API は、インバウンド・リソース・アダプタから操作コンソールへ、重要またはクリティカルな（あるいはその両方の）状態変更について通信するためのもので、これによって、適時に手動操作を行うことが可能になります。これには、たとえば、データベースや他の EIS 機能の再起動などがあります。

Oracle Enterprise Service Bus でサポートされない機能

Oracle Enterprise Service Bus 10.1.3 でサポートされない機能は次のとおりです。

- 暗黙的な相関
- 大きなペイロードのストリーミング
- インバウンドおよびアウトバンドのリクエスト / リプライ
- Dynamic JCA パートナ・リンク
- ESBListenerImpl のクラスタ化されたインスタンス間での透過的フェイルオーバー
- インバウンド XPath フィルタリング

5.4.2.3 アダプタ・サービスを介した Oracle Enterprise Service Bus への接続性

Oracle Application Server アダプタでは、社内のほぼすべてのデータ・ソースに双方向でリアルタイムなデータ・アクセスが可能です。

アダプタは、サポートするソース・アプリケーションのイベントをリスニングまたはポーリングします。イベントをリスニングする場合、アダプタは、イベントをアダプタにプッシュするよう構成されたアプリケーションのリスナーとして登録されます。また、アダプタは Oracle Enterprise Service Bus で必要とされるイベントについて、データベースやファイルなどのバックエンド・アプリケーションのポーリングもできます。

Oracle Enterprise Service Bus にアダプタを（ウィザードを使用して）登録することにより、外部データ・ソースを Oracle Enterprise Service Bus に統合し、最終的には相互に統合します。

Oracle Enterprise Service Bus サーバーでは、現在、次の表に記載された Oracle アダプタをサポートしており、これにより、インバウンドおよびアウトバウンドのアダプタ・サービスをそれぞれ定義できます。インバウンド・アダプタ・サービスでは、外部データ・ソースからデータを受信し、これを XML メッセージに変換します。アウトバウンド・アダプタ・サービスでは、XML メッセージを所定のアダプタのネイティブ・フォーマットに変換して、ターゲット・アプリケーションへデータを送信します。

アダプタ・サービス	説明
AQ アダプタ・サービス	Oracle Advanced Queuing のシングル・キューまたはマルチコンシューマ・キューからのメッセージを送受信します。
ファイル・アダプタ・サービス	ローカル・ファイル・システムのファイルからのメッセージを送受信します。
FTP アダプタ・サービス	リモートのファイル転送プロトコル（FTP）サーバーのファイルからのメッセージを送受信します。

アダプタ・サービス	説明
データベース・アダプタ・サービス	Oracle Database の表から抽出したメッセージ、またはストアド・プロシージャを実行して作成したメッセージを送受信します。
JMS アダプタ・サービス	JMS キューまたはトピックからのメッセージを送受信します。
MQ アダプタ・サービス	IBM の MQ キューまたはトピックからのメッセージを送受信します。
Oracle Application アダプタ・サービス	Oracle E-Business Suite インタフェースからのメッセージを送受信します。

インバウンド・アダプタ・サービスを除き、Oracle Enterprise Service Bus サービスとして作成するアウトバウンド・アダプタ・サービスやルーティング・サービスなどのサービスはすべて、SOAP サービスとして、詳細の構成の必要なく自動的に作成されます。インバウンド・アダプタ・サービスの場合は、手動で SOAP サービスとして構成する必要があります。Oracle Enterprise Service Bus では、構成プロセスの手順に沿ったウィザードが提供されます。

6

アダプタのライフ・サイクル管理

Oracle Application Server アダプタは、J2EE Connector Architecture（J2CA）1.5 標準に基づいており、Oracle Containers for J2EE（OC4J）内にデプロイされます。Oracle Application Server アダプタのライフ・サイクルは、Business Process Execution Language for Web Services（BPEL）Process Manager によって決まります。これらのアダプタは、アダプタ・フレームワーク・コンポーネントを介して BPEL Process Manager に統合されます。

この章の内容は、次のとおりです。

6.1 Oracle Application Server アダプタのインストール

テクノロジ・アダプタおよび OracleAS Adapter for Oracle Applications は、Oracle BPEL Process Manager のインストールの一部として組み込まれています。これらのアダプタは、スタンドアロン OC4J と中間層の両方のデプロイをサポートしています。パッケージ・アプリケーション・アダプタおよびレガシー・アダプタは、OracleAS Adapters CD の一部として使用できます。これらのアダプタは、中間層デプロイのみをサポートしています。

6.2 アダプタの起動と停止

Oracle Application Server アダプタは、J2CA 1.5 リソース・アダプタとしてデプロイされます。したがって、アダプタを起動および停止するには、すべてのリソース・アダプタが、`start(BootstrapContext)` メソッドと `stop` メソッドを SPI インタフェースの一部として実装している必要があります。Oracle Application Server アダプタは、それらのアダプタを使用する BPEL プロセスが J2CA アウトバウンド相互作用を開始したときに起動されます。アダプタは、BPEL プロセス自体がインバウンド相互作用のためにロードされるとき、またはアダプタが BPEL プロセスにイベントをパブリッシュするときにも起動されます。

アダプタの起動後にアダプタを停止する唯一の方法は、OC4J を停止することです。このリリースでは、アダプタ・フレームワークは J2CA 1.5 コンテナの一部として機能します。

6.3 アダプタの物理的なデプロイ

Oracle Application Server アダプタは、OC4J コンテナ内に J2CA 1.5 リソース・アダプタとしてデプロイされます。アダプタは、Java アーカイブ（JAR）形式を使用してリソース・アダプタ・アーカイブ（`RAR`）ファイルとしてパッケージ化されています。アダプタを物理的にデプロイするには、`RAR` ファイルを使用すること、および基礎となる OC4J または中間層プラットフォームにアダプタをコネクタとして登録することが必要です。`RAR` ファイルには、`ra.xml` ファイルが含まれています。このファイルは、リソース・アダプタに関するデプロイ固有の情報が格納されるデプロイメント・ディスクリプタ XML ファイルです。また、`RAR` ファイルには、OC4J とリソース・アダプタ間の規約に関する宣言情報も含まれています。

アダプタは、`.rar` ファイル内の `ra.xml` ファイルに加え、`oc4j-ra.xml` テンプレート・ファイルをパッケージ化します。`oc4j-ra.xml` ファイルは、リソース・アダプタの `ConnectorFactory` オブジェクト（論理的なデプロイ）の定義に使用されます。この `oc4j-ra.xml` ファイルは、リソース・アダプタ用の OC4J 固有のデプロイメント・ディスクリプタです。このファイルには、リソース・アダプタを OC4J にデプロイするためのデプロイ構成が含まれており、これには、リソース・アダプタのデプロイメント・ディスクリプタに指定されているバックエンド・アプリケーションの接続情報、使用する Java Naming and Directory Interface（JNDI）の名前、接続プーリング・パラメータ、リソース・プリンシパル・マッピング・メカニズムおよび構成が含まれます。

例 6-1 スタンドアロン OC4J インストール

この例では、BPEL Process Manager のスタンドアロン OC4J インストールの一部として、テクノロジおよび OracleAS Adapter for Oracle Applications を物理的にデプロイする方法を示します。スタンドアロン OC4J インストールでは、次のコマンドを使用して JCA 1.5 リソース・アダプタをデプロイします。

```
java -jar $ORACLE_
HOME/integration/orabpel/system/appserver/oc4j/j2ee/home/admin.jar
ormi://localhost oc4jadmin welcome1 -deployConnector -file myAdapter.rar -name
myAdapter
```

スタンドアロン OC4J インストールでは、次のコマンドを使用して JCA 1.5 リソース・アダプタを削除します。

```
java -jar $ORACLE_
HOME/integration/orabpel/system/appserver/oc4j/j2ee/home/admin.jar
ormi://localhost oc4jadmin welcome1 -undeployConnector -name myAdapter
```

MANIFEST.MF、oc4j-ra.xml および ra.xml ファイルは、コネクタ・フォルダにも格納されます。

サンプルの oc4j-ra.xml ファイルを次に示します。

```
<oc4j-connector-factories>
      <connector-factory>
            <config-property name="maxReadConnections" value="1"/>
            <config-property name="usesExternalConnectionPooling" value="false"/>
            <config-property name="dataSourceName" value=""/>
            <config-property name="usesExternalTransactionController" value="false"/>
            <config-property name="usesNativeSequencing" value="true"/>
            <config-property name="sequencePreallocationSize" value="50"/>
            <config-property name="tableQualifier" value=""/>
      </connector-factory>

    <!-- A connection used by the samples/tutorials to run out of the
         box using the olite repository.
         Also a template olite connection.
    -->
    <connector-factory location="eis/DB/BPELSamples" connector-name="Database Adapter">
        <config-property name="driverClassName" value="oracle.lite.poljdbc.POLJDBCDriver"/>
        <config-property name="platformClassName"
value="oracle.toplink.internal.databaseaccess.OraclePlatform"/>
        <config-property name="connectionString" value="jdbc:polite4@localhost:100:orabpel"/>
        <config-property name="userName" value="system"/>
        <config-property name="password" value="any"/>
        <config-property name="minConnections" value="1"/>
        <config-property name="maxConnections" value="5"/>
        <config-property name="minReadConnections" value="1"/>
        <config-property name="maxReadConnections" value="5"/>
        <config-property name="usesExternalConnectionPooling" value="false"/>
        <config-property name="dataSourceName" value=""/>
        <config-property name="usesExternalTransactionController" value="false"/>
        <config-property name="usesNativeSequencing" value="true"/>
        <config-property name="sequencePreallocationSize" value="50"/>
            <config-property name="tableQualifier" value=""/>
```

注意： 中間層インストールでは、dcmctl ユーティリティを実行して、J2CA 1.5 リソース・アダプタをデプロイします。次のコマンドを使用してください。

```
cd $ORACLE_HOME/dcm/bin
./dcmctl deployApplication -f myAdapter.rar -a myAdapter -co OC4J_
SOA
```

6.4 アダプタの論理的なデプロイ

アダプタの論理的なデプロイとは、oc4j-ra.xml デプロイメント・ディスクリプタ・ファイルでの ConnectionFactory オブジェクトの作成を意味します。接続情報を追加して JNDI 名に割り当てるには、リソース・アダプタの対応する oc4j-ra.xml ファイルを編集します。また、必要なアダプタの oc4j-ra.xml ファイルを手動で検索して編集する必要があります。oc4j-ra.xml ファイルには、アダプタのランタイム接続パラメータが含まれています。

たとえば、OracleAs Adapter for Database を論理的にデプロイするには、次の場所にある oc4j-ra.xml ファイルを編集します。

$ORACLE_HOME/integration/orabpel/system/appserver/oc4j/j2ee/home/application-deployments/default/DbAdapter/oc4j-ra.xml.

6.5 アダプタ・ログの表示

次の Oracle Application Server アダプタのログを表示できます。

- テクノロジ・アダプタおよび OracleAS Adapter for Oracle Applications: これらのアダプタは、アダプタ・フレームワークの `LogManager` インタフェースを実装します。このインタフェースを使用して、アダプタ・ログを BPEL ドメイン・ログにリダイレクトします。アウトバウンド相互作用とインバウンド相互作用の両方について、ログ・ファイルが BPEL ドメイン・ログにリダイレクトされます。

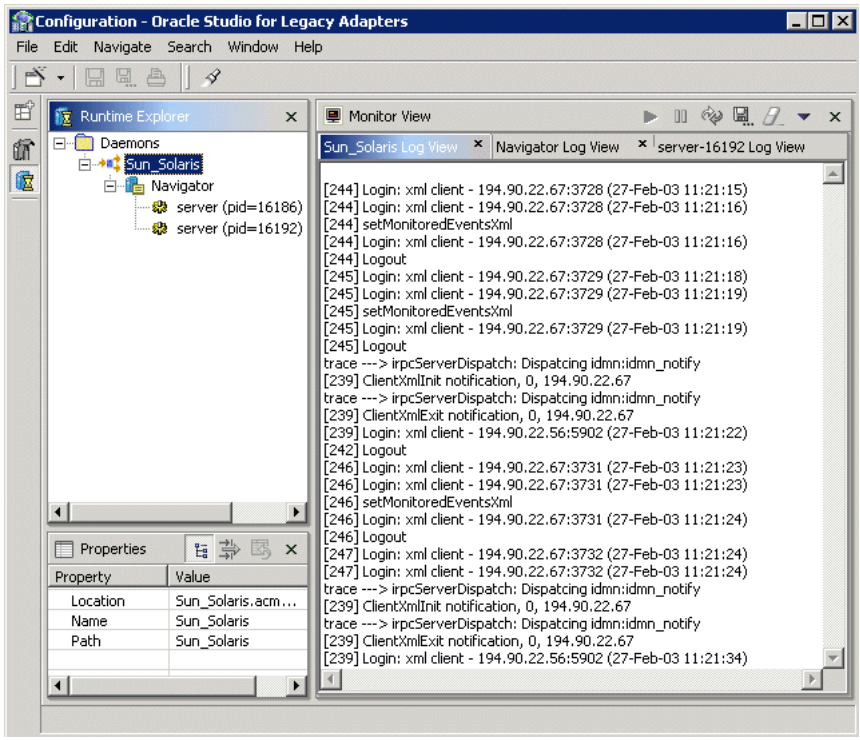
  BPEL ドメイン・ログは、次の場所にあります。

  $ORACLE_HOME`/integration/orabpel/domains/default/logs/domain.log`.

  注意： BPEL コンソールからアダプタ・ログを表示することもできます。

- パッケージ・アプリケーション : これらのアダプタは、J2CA 1.5 標準の一部ではないため、`LogManager` インタフェースを実装しません。したがって、これらのアダプタ・ログ出力は、`opmn/logs` ディレクトリにリダイレクトされます。アウトバウンド相互作用の場合、ログは opmn ログに出力されます。一方、インバウンド相互作用の場合、ログは BPEL ドメイン・ログにリダイレクトされます。

- レガシー・アダプタ : J2CA リソース・アダプタに加え、レガシー・アダプタは、メインフレーム・アプリケーションおよびデータ・ストアと通信するための複数のネイティブ・アダプタからなる Oracle Connect で構成されています。Oracle Connect のログは、Oracle Studio を使用して表示できます。Oracle Studio には、メインフレーム・アダプタ設計時ツールと Oracle Connect 管理ツールがあります。Oracle Connect では、デーモン・ログ、ワークスペース・ログ、サーバー・プロセス・ログなど、様々なタイプのログが生成されます。次の図は、デーモン・ログ・モニターを示しています。このモニターには、デーモンにログインおよびログアウトしているクライアントなど、クライアントとデーモン間のアクティビティが表示されます。



6.6 アダプタ・ヘッダーの使用

アダプタは、基礎となるバックエンド操作固有のプロパティをヘッダー要素として公開し、ビジネス・プロセス内でこれらの要素を操作できるようにします。たとえば、OracleAS Adapter for Advanced Queuing のヘッダー要素は、1 つのビジネス・プロセス内に 2 つの Advanced Queuing メッセージを関連付けるための相関 ID やメッセージ ID など、ヘッダーのプロパティを公開します。

6.7 アダプタのトレース・レベルの設定

次の Oracle Application Server アダプタのトレース・レベルを設定します。

- テクノロジ・アダプタおよび OracleAS Adapter for Oracle Applications: BPEL コンソールを使用して、テクノロジ・アダプタおよび OracleAS Adapter for Oracle Applications のトレース・レベルを設定します。BPEL コンソールをオープンした後、**「BPEL ドメインの管理」**をクリックし、**「ロギング」**タブを選択します。アウトバウンド相互作用には、`default.collaxa.cube.ws` プロパティを設定します。インバウンド相互作用には、`default.collaxa.cube.activation` プロパティを設定します。
- パッケージ・アプリケーション・アダプタ : アウトバウンド相互作用には、`oc4j-ra.xml` ファイル内のパッケージ・アプリケーション・アダプタの `Loglevel` プロパティを設定します。
- レガシー・アダプタ : Oracle Connect（メインフレーム・サーバー）のトレース・レベルは、Oracle Studio を使用して設定できます。

6.8 JDeveloper 内のアダプタ・デプロイ検証

デプロイ中（JDev または obant）、J2CA バインディングを含む WSDL は、J2CA WSDL 検証サービスにより精査されます。WSDL 検証サービスでは、J2CA バインディングを含む WSDL のみが選択されます。

検証ステップは次のように実施されます。

- WSDL（JCA）バインディングの整合性
- JDev または managedConnectionFactory/ConnectionSpec（`jca:service` で提供されている場合）を介してデプロイされる場合の JNDI ロケーションの検証
- `jca:operation` で指定されている InteractionSpec または ActivationSpec の存在
- `jca:operation` で InteractionSpec または ActivationSpec にバインドされているプロパティ名と値すべての検証
- jca:binding または `pc:inbound_binding` で指定されている XML レコード・コンバータすべての検証
- 最後に、InteractionSpec または ActivationSpec で DeploymentValidation インタフェースが実装される場合には、その `validate()` メソッドが起動します。

6.9 XML 検証の有効化

BPEL サーバー・ドメインの XML 検証は、BPEL コンソールを介して、または `PartnerLink` レベルで有効化できます。インバウンドおよびアウトバウンドの XML 検証は、BPEL Process Manager で有効化できます。

6.10 XML データ構造の記述

Oracle Application Server アダプタのレコード実装は、XMLRecord です。アダプタ相互作用はすべて XMLRecord で開始します。各 JCA レコードは、`oracle.tip.adapter.api.record.XMLRecord` の実装であることが必要です。XMLRecord の各インスタンスには、1 つまたは 2 つの RecordElement のインスタンス（1 つは必須のペイロード RecordElement、もう 1 つはオプションのヘッダー RecordElement）があります。RecordElements にはデータが格納されています。また、各 RecordElement には、UTF-8 エンコードの XML 文字列または不透明（Opaque）なバイナリ・バイト・ストリームの BLOB データが 1 つ含まれています。

XMLRecord は、次のメソッドで構成されます。

- `getHeaderRecordElement`: ヘッダー RecordElement を取得します。
- `getPayloadRecordElement`: ペイロード RecordElement を取得します。
- `setHeaderRecordElement`: XMLRecord のヘッダー・レコード要素を設定します。
- `setPayloadRecordElement`: XMLRecord のペイロード・レコード要素を設定します。

6.11 oc4j-ra.xml でのパスワードの暗号化

`orabpel¥bin` ディレクトリの `encrypt` コマンド（`encrypt.sh` または `encrypt.bat`）を使用して、`oc4j-ra.xml` ファイルのパスワードを暗号化します。

各 JCA リソース・アダプタ実装は、このパスワードを復号化するために、適切なアダプタ・フレームワークの復号化機能を使用する必要があります。アダプタで復号化できないと、`oc4j-ra.xml` エントリの暗号化が意味をなしません。

6.12 エラーの管理

アダプタおよびその基盤となっているアダプタ・フレームワーク（AF）・ライブラリでは、次のタイプの例外がスローされます。

- WSIFException: インバウンドおよびアウトバンド両方の相互作用に対する JCA アーティファクトが、アダプタ・サービスを定義する WSDL に取得されます。実行時に、AF の JCA WSIF プロバイダ・コンポーネントにより、JCA アーティファクトについて WSDL が解析され、アウトバウンド相互作用のための InteractionSpec が作成されます。InteractionSpec クラスがクラスパス内に特定できない場合、またはインスタンス化が失敗した場合は、AF から `WSIFException` がスローされます。
- ActivationException: BPEL プロセスで JCA インバウンド相互作用に関連するアダプタ・エンドポイントのアクティブ化に失敗した場合、`ActivationException` がスローされます。
- PCRetriableResourceException: アダプタでは、一時的な接続エラーに対し、`PCRetriableResourceException` がスローされます。このエラーは、アウトバウンド相互作用の場合にはリカバリ可能です。`bpel.xml` でリトライ・ポリシーを定義することにより、PCRetriableResourceException 例外の再試行が可能となります。
- ResourceException: `ResourceException` 例外は、すべてのケースについてスローされます。

6.13 接続エラーの処理

テクノロジおよびOracleAS Adapter for Oracle Applicationsのアダプタは、次の相互作用に対する接続エラーを処理できます。

- アウトバウンド相互作用 : テクノロジ・アダプタおよびOracleAS Adapter for Oracle Applicationsでは、一時的な接続エラー（リカバリ可能な接続エラー）に対して、`oracle.tip.adapter.api.exception.PCRetriableResourceException`が発生します。たとえば、データベース・リスナーが起動していないために、接続エラーが発生する可能性があります。

  実行する再接続の最大試行回数を`bpel.xml`ファイルに定義できます。このファイルの`PartnerLinkBinding`パラメータの下に、再接続を試行するためのパラメータを次のように指定します。

```
<BPELSuitcase>
<BPELProcess ...>
  <partnerLinkBindings>
     <partnerLinkBinding name="myOutboundPartnerLink">
     <property name="wsdlLocation">Outbound.wsdl</property>
     <property name="retryInterval">10</property>
     <property name="retryMaxCount">30</property>
  </partnerLinkBinding>
```

  この例の再接続パラメータ設定では、10秒ごとに再接続を試行し、最大30回の再接続を試行するように、BPELランタイムを指定しています。最大回数まで試行すると、BPELプロセスの`RemoteFault`例外が発生します。

  他のすべての例外はリカバリ不可で、`BindingFault`例外がBPELプロセスにスローされる結果となります。

  注意 : OracleAS Adapter for Oracle Applicationsを除くパッケージ・アプリケーション・アダプタおよびレガシー・アダプタでは、接続例外エラーが戻されます。これらのアダプタについては、サード・パーティの例外を`PCRetriableResourceException`（`RemoteFault`）か、または`PCResourceException`（`BindingFault`）のいずれかに変換するException Filter（Javaクラス）を準備する必要があります。この例外フィルタは、次のように`jca:binding`要素で宣言します。

```
<jca:binding
ExceptionFilter="oracle.tip.adapter.sample.SampleExceptionFilter"/>
```

  このフィルタ（Javaクラス）にはインタフェース`oracle.tip.adapter.api.exception.ExceptionFilter`を実装する必要があります。

- インバウンド相互作用 : テクノロジ・アダプタ、Oracle ASアダプタおよびレガシー・アダプタは、イベント受信のためのバックエンド・アプリケーション接続用のポーリング・モデルをサポートしています。リカバリ不能な接続エラーの場合には、アダプタは旧接続を再使用し、BPELプロセスにアラートまたは通知を送信します。さらに、アダプタはインバウンド・エンドポイントおよび関連するBPELプロセス・インスタンスを停止します。このインバウンド相互作用接続エラーはログに記録され、BPELコンソールで表示できます。

6.14 アダプタ・データ・エラーの処理

次の相互作用の際に発生する可能性があるエラーを処理できます。

- アウトバウンド相互作用 : J2CA アダプタは、アウトバウンド相互作用に対して `ResourceAdapter` 例外をスローします。データの修正およびメッセージの再生は、BPEL プロセス内で処理される必要があります。
- インバウンド相互作用 : 各 J2CA アダプタは、終了状態でのフェイルオーバー BPEL プロセスに加え、1 つ以上のメッセージ拒否ハンドラを関連付けることができます。`bpel.xml` では、これらのメッセージの処理方法を次のように構成できます。
  - エラーを特定のディレクトリに出力する
  - エラーを Advanced Queue（AQ）キューに出力する
  - エラーをエラー・ハンドラ BPEL プロセスに送信する
  - WSIF サービスをコールする

6.15 メッセージの順序の記述

メッセージの順序付けは、BPEL プロセスを同期させることにより達成できます。これは BPEL エンジンにメッセージをパブリッシュするリソース・アダプタのスレッドが、BPEL インスタンス全体を実行するために使用されることを意味します。

非同期 BPEL プロセスでは、受信したメッセージは BPEL エンジンのワーカー・スレッドのプールで処理されます。この場合、メッセージの順序は保証されません。ほとんどの場合、BPEL インスタンスの期間は他の BPEL インスタンスの期間と異なるためです。

通常、JCA リソース・アダプタ・ウィザードでは、一方向（非同期）、つまり、入力メッセージのみを操作する WSDL が作成されます。ウィザードで生成された WSDL を手動で変更して、インプットおよびアウトプット・メッセージのあるリクエスト - レスポンス（同期）・タイプの WSDL にする必要があります。

次の例を考慮に入れて、通常は生成されないレスポンス（アウトプット）・メッセージ用の XML スキーマ・タイプを作成してください。

```
< タイプ >
  <schema xmlns="http://www.w3.org/2001/XMLSchema"
          targetNamespace="http://acme.com/adapterService/">
    <import namespace="http://TargetNamespace.com/adapterService/type"
            schemaLocation="adapterTypes.xsd" />
.
.
.
      <element name="empty">
        <complexType/>
      </element>
```

XML スキーマ・タイプを作成した後、次のように WSDL メッセージを（アダプタ WSDL ファイル内に）定義します。

```
<message name="ignoreMessage">
  <part name="empty" element="tns:empty"/>
</message>
```

次に、例に示すように、インバウンド WSDL 操作にアウトプット・メッセージを追加します。

```
<portType name="Receive_ptt">
  <operation name="Receive">
    <input message="tns:payloadMessage"/>
    <output message="tns:ignoreMessage"/>
  </operation>
</portType>
```

次に、例に示すように、バインディング・セクションにアウトプット要素を追加します。

```
<binding name="Receive_binding" type="tns:Receive_ptt">
  <pc:inbound_binding />
  <operation name="Receive">
    <jca:operation .../>
    <input/>
    <output/>
  </operation>
</binding>
```

WSDL はこれで OK です。

次に、例に示すように、BPEL プロセスのビジネス・プロセス・ロジックの最後に再試行アクティビティを追加します。

```
<variables>
  <variable name="ignore" messageType="ns1:ignoreMessage"/>
  .
  .
  .
<sequence name="main">
  <receive partnerLink="ReceivePL" portType="ns1:Receive_ptt"
           operation="Receive"
           variable="Receive_1_Read_InputVariable"
           createInstance="yes"/>
           .
           .
           .
           <!-- processing -->
  <invoke partnerLink="DB_Insert" portType="ns1:DB_Insert_ptt"
          operation="InsertCust" inputVariable="NewCust"/>

<reply partnerLink="ReceivePL" portType="ns1:Receive_ptt"
         operation="Receive" variable="ignore"/>
         .
         .
         .
         <!-- optionally more processing -->
</sequence>
```

前述の例でリソース・アダプタがマルチスレッド・インバウンドをサポートしている場合には、1 つのスレッドのみを使用するように構成または調整する必要があります。複数スレッドでは、確率論的な期間によって、メッセージの順序が保証されなくなるためです。

6.16 メッセージ拒否ハンドラの記述

メッセージ拒否ハンドラは、次のことを行うように構成できます。

- トランスレーション・エラーの処理
- 修正の実行
- BPEL プロセスへのアラートまたは通知メッセージの発行

関連する PartnerLink アクティビティ下で bpel.xml ファイル内に Message Rejection ハンドラを指定します。さらに、メッセージ拒否ハンドラをアクティブ化エージェント・プロパティの一部として定義する必要があります。メッセージ拒否ハンドラの 4 つのタイプを次に示します。

- ファイル・システム・ベースのメッセージ拒否ハンドラ : ファイル・システム・ベースのメッセージ拒否ハンドラは、次のファイル名パターンを使用して、設定されたディレクトリに不正メッセージを書き込みます。

  ```
  INVALID_MSG_ + <process-name> + <operation-name> + <current-time>
  ```

  ファイル・システム・ベースの拒否ハンドラを指定する構文は次のとおりです。

  ```
  <property name="rejectedMessageHandlers">
      file://<directory-path>
  </property>
  ```

  ファイル・システム・ベースの拒否ハンドラを指定する例を次に示します。

  ```
  <property name="rejectedMessageHandlers">
      file://C:/orabpel/domains/default/rejectedMessages
  </property>
  ```

- RAW Oracle アドバンスト・キュー・ベースのメッセージ拒否ハンドラ : RAW Oracle アドバンスト・キュー・ベースのメッセージ拒否ハンドラでは、ユーザーは拒否の格納先として Oracle RDBMS RAW AQ キューを指定できます。

  RAW Oracle アドバンスト・キュー・ベースのメッセージ拒否ハンドラを指定する例を次に示します。

  ```
  <property name="rejectedMessageHandlers">
  queue://jdbc:oracle:thin:@<db-host>:<tns-port>:<sid>|<user>/<password>|<queue-n
  ame>
  </property>
  ```

  「:」および「|」は、コード内で、この例に示されているとおりの位置に入力してください。

- BPEL プロセスのメッセージ拒否ハンドラ : BPEL プロセスのメッセージ拒否ハンドラは、指定されたエラー処理 BPEL プロセスに不正メッセージを送信します。

  BPEL プロセスのメッセージ拒否ハンドラを指定する構文は次のとおりです。

  ```
  <property name="rejectedMessageHandlers">
  bpel://<bpel-domain[:<password>]>|<process-name>|<operation-name>|<input-messag
  e-partname>
  </property>
  ```

注意 : 前述の構文中の [:<password>] は encrypt.bat/encrypt.sh ユーティリティを使用して暗号化できます。

このように、プロセスは受信操作の（WSDL および BPEL ソースに関する）選択によって定義できます。唯一の制約は、この拒否ハンドラに送信されるメッセージの WSDL メッセージ・タイプに関する制約のみです。このタイプは、RejectedMessage タイプとして宣言されている必要があります。この宣言を行うには、xmllib に常駐する WSDL RejectionMessage.wsdl をインポートします。このファイルで、次の例に示すようにメッセージを定義します。

```
<message name="RejectionMessage">
     <part name="message" element="err:RejectedMessage"/>
</message>
```

次の例に示す URL を使用して、他の WSDL から xmllib WSDL をインポートできます。

```
<import namespace="http://xmlns.oracle.com/pcbpel/rejectionHandler"
location="http://localhost:9700/orabpel/xmllib/jca/RejectionMessage.wsdl"/>
```

BPEL プロセスには、インポートをこの例に示すとおりにのみ含めることができます。ポート・タイプは次のように参照されます。

```
<definitions ...
xmlns:rej="http://xmlns.oracle.com/pcbpel/rejectionHandler"
   <portType name="MyRejectionHandlerPortType">
      <operation name="myHandleRejectionOperation">
      <input message="rej:RejectionMessage"/>
      </operation>
   </portType>
```

- WSIF ベースの拒否ハンドラ : WSIF ベースのメッセージ拒否ハンドラでは、JCA、EJB、JMS、HTTP および Java などの WSIF WSDL のすべてのタイプを設定できます。WSIF を介して不正メッセージ・ハンドラとして達することの可能などのようなタイプのサービスも設定できます。

  WSIF ベースのメッセージ拒否ハンドラを指定する構文は次のとおりです。

  ```
  <property name="rejectedMessageHandlers">
      wsif://<wsif-wsdl-location>|<operation-name>|<input-message-part-name>
  </property>
  ```

  次の例のようにして、WSIF ベースのメッセージ拒否ハンドラを指定できます。

  ```
  <property name="rejectedMessageHandlers">
  wsif://file:/C:/orabpel/samples/test/ErrorTest/FileAdapterWrite.wsdl|write|mess
  age
  </property>
  ```

  WSIF ベースのメッセージ拒否ハンドラには、メッセージ・タイプに関して、BPEL プロセスの拒否ハンドラと同じ制約があります。BPEL プロセスの拒否ハンドラの項を参照してください。

6.17 アダプタでメッセージの欠落のないことを確認する方法の記述

この項では、アダプタでメッセージの欠落のないことを確認する方法、およびBPELデハイドレーション・ストアからメッセージをリカバリする方法を記述します。

BPELエンジンは常に配信を確認するよう構築されています。デハイドレーション・ポイントは、reveive、pickおよびwaitの前と、replyおよびinvokeの後に、それぞれ設置されます。デフォルトでは起動後のデハイドレーションは遅延しますが、チューニング・プロセス・パラメータを介して管理できます（BPELチューニング・ガイドのidempotent設定オプションを参照してください）。

JCAリソース・アダプタによって、BPELエンジンの及ぶ範囲が拡張され、配信保証も特定の方法により確実に維持されます。

トランザクション・アダプタにより、EISを1フェーズまたは2フェーズ・コミット（ローカル・トランザクション、あるいは、グローバルまたは分散トランザクション）に加えることができます。これらのコミットは、アダプタ（インバウンド）またはBPELエンジン（アウトバウンド）により制御されます。非トランザクション・アダプタには、トランザクション・セマンティクスを使用せずに配信を確実にするための独自のスキームが実装されています。

6.17.1 ローカル・トランザクションおよびグローバル（XA）・トランザクション

BPEL PM 10.1.3のトランザクション・アダプタでは、JCA 1.5 XA規約を介して、グローバル（2フェーズ・コミット）・トランザクションをサポートしています。これには、Oracle Applications、Oracle Database、Advanced Queuing、JMSおよびMQSeries用のアダプタが含まれます。非トランザクション・アダプタには、ファイル・アダプタおよびFTPアダプタが含まれます。

6.17.2 インバウンド・トランザクション

非同期BPELプロセスの場合、トランザクション・アダプタはグローバルJTAトランザクションを開始してからインバウンド・メッセージをBPELに送信します。このインバウンド・メッセージは、BPELエンジンの配信サービスを介してデハイドレーション・ストアのキューに入ります。制御がアダプタに戻ると、アダプタはJTAトランザクションをコミットします。このようにして、次の一連のアクションを最小の作業ユニットとして実行します。

1. インバウンド・アダプタ・エンドポイントからのメッセージの削除をコミットします（表およびキューなど）。
2. このメッセージのデハイドレーション・ストアへの挿入をコミットします。

この処理中に何らかの失敗があった場合、アクション1および2の両方がロールバックされることが保証されています。

アダプタからデハイドレーション・ストアへのメッセージ配信が成功した後は、そのインバウンド・メッセージは次のBPELプロセスのデハイドレーション・ポイントが発生するまで（たとえば、最初の起動まで）、ストア内に残ります。次のデハイドレーション・ポイントで、受信アクティビティからのメッセージは配信サービスから削除されます。これらのチェックポイント間のアクティビティはすべてJTA（グローバル）トランザクションの一部として扱われます。2つのデハイドレーション・ポイント間でBPELサーバーに障害が発生した場合は、トランザクション全体がロールバックされて、状態が保証されます。最新のメッセージは、次に使用可能となったBPELサーバーにより再実行されます。これらはすべて表面的に見えることなく行われます。

同期プロセスの場合、アダプタにより開始されたグローバル・トランザクションでは、メッセージ配信およびBPELプロセスの実行が最初のリプライ・アクティビティ（通常プロセス・フローの最後に配置）までにわたります。

6.17.3 アウトバウンド・トランザクション

トランザクション・アダプタの場合、アウトバウンド JCA 相互作用（invoke アクティビティ）は、グローバル BPEL JTA（ejb）トランザクションで精査されます。これは JCA アダプタの起動を含むすべての BPEL アクティビティがグローバル・トランザクションの一部となることを意味し、そのため、すべてのアクティビティは、コミットされるか、またはエラーが発生した場合はロールバックされるかのどちらかになります。

たとえば、BPEL プロセスでは、（データベース・アダプタを起動する）異なる invoke アクティビティを介して、（異なるデータベース上の）複数の表にデータを挿入できます。BPEL インスタンスがほぼ終了すると、JTA トランザクションはコミットされます。この時点でのみデータベースの挿入操作がコミットされます。BPEL インスタンスの実行中にエラーが生じた場合は、すべてのアクティビティ（およびデータベース操作）は最後のデハイドレーション・ポイントまでロールバックされます。

6.18 アダプタ内での BPEL クラスタ化サポート

アダプタ・フレームワークでは、インバウンド・アダプタ・サービスのアクティブ・フェイルオーバーがサポートされます。これを実現するには、特定の JCA アクティブ化エージェントにプロパティを追加します（bpel.xml 内）。次に例を示します。

```
<activationAgents>
  <activationAgent className="..." partnerLink="MyInboundAdapterPL">
    <property name="clusterGroupId">myBpelCluster</property>
```

クラスタ内の各 BPEL PM サーバー（JVM）が TCP/IP のサブネットの境界を越えて配置されている場合は、属性 `clusterAcrossSubnet=true` を追加する必要があります。

クラスタ・グループ内では、同じアダプタ（ファイル・アダプタなど）のアクティブ化エージェントが（特定のパートナ・リンクについて）複数アクティブ化されると、そのクラスタ内でアクティブになっているアダプタ・フレームワークのすべてのインスタンスで、暗黙的かつ自動的に検出されます。実際にメッセージの読取りまたはパブリッシュを開始するために許可されるアクティブ化は、1 つのみです。アダプタ・フレームワーク・インスタンスによって、複数あるアクティブ化のうち、どれをプライマリとするかについてはランダムに、1 つが選択されます。クラスタ内のその他のアクティブ化（インスタンス）は、JCA リソース・アダプタ上で実際に EndpointActivation を起動させずに、ホット・スタンバイ状態で開始されます。

プライマリのアクティブ化対象がいずれかのポイントで反応しなくなった場合、つまり、手動で非アクティブ化されたり、クラッシュまたは終了した場合は、クラスタ・グループの残りのアダプタ・フレームワーク・メンバーのうちの 1 つが即時にこれを検出し、プライマリのアクティブ化対象をスタンバイのアクティブ化エージェントのうちの 1 つに割り当てます。この機能では、実装の基底部分で JGroups の clusterGroupId プロパティを使用しています。

6.19 アダプタ・サービスのデプロイ

アダプタ・サービスは、JDeveloper と BPEL コンソールを使用して、2 つの方法でデプロイできます。また、BPEL プロセスをデプロイすることでもアダプタ・サービスがデプロイされます。アダプタ・サービスは、BPEL コンソールを使用して削除できます。

6.20 バッチ化およびバッチ分割化のサポート

バッチ化およびバッチ分割化機能は、OracleAS Adapter for Files、OracleAS Adapter for FTP および OracleAS Adapter for Databases のみでサポートされます。OracleAS Adapter for Files および OracleAS Adapter for FTP は、単一の大きなファイルを複数のバッチに分割できるリーダーで構成されます。設計時の構成でバッチ・サイズを指定する必要があります。また、アダプタには、一連のメッセージを単一のファイルにバッチ化するライターが含まれています。

OracleAS Adapter for Databases は、一連の表をポーリングしてイベントを検出するパブリッシュ・コンポーネントで構成されます。このコンポーネントは、一度に 1 つのレコードまたは一度に複数のレコードを BPEL プロセスに呼び出すことができます。

6.21 リポジトリの移行

アダプタ構成ウィザードで生成されるすべてのアダプタ・サービス WSDL は、JNDI 名への参照を持ちます。この参照は、アダプタのデプロイメント・ディスクリプタである `oc4j-ra.xml` 内で、`jca:address` 要素のロケーション属性を介して定義されます。これが、開発環境から本番環境のためのテスト環境へ移行する際のキーとなります。`oc4j-ra.xml` ファイルを更新して、開発、テストおよび本番の 3 つの環境すべてにおいて同じ JNDI 名を持つようにする必要があります。再試行間隔や再試行回数など、デプロイ時のプロパティに対して値を指定し、その後、テスト環境や本番環境に再デプロイする必要があります。`oc4j-ra.xml` では、エンドポイントが開発 EIS、テスト EIS または本番 EIS として識別されます。たとえば、データベース・アダプタ・サービス・ウィザードを介して実行する場合に、`createCustomer` サービス用の JNDI 名として `eis/DB/custStore` を指定したとします。このアダプタ・サービスを使用して BPEL プロセスをモデリングした後は、これを変更せずに開発、テストまたは本番環境にデプロイする必要があります。ただし、これを実行する前に、各環境において、正しい EIS インスタンスを示す `eis/DB/custStore` に対応した JNDI エントリがあることを確認してください。

索引

ら

れ