Oracle® Application Server

Adapter Concepts Guide

10*g* Release 3 (10.1.3.1.0)

B31005-01

September 2006

ORACLE®

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

B31005-01

Primary Author: Sheela Vasudevan

Contributing Authors: Bo Stern, Meera Srinivasan, Maneesh Joshi, Prasad Dixit-Hardikar

Contents

5 Adapter Integration with Oracle Application Server Components

6 Adapter Life-Cycle Management

Preface

This preface contains the following topics:

Audience

Oracle Application Server Adapter Concepts Guide is intended for those users who want to learn about the concepts of the following adapters:

- Technology
- Packaged Application
- Legacy

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

For more information, see the following documents in the Oracle Other Product One Release 7.0 documentation set or in the Oracle Other Product Two Release 6.1 documentation set:

- *Oracle Application Server Administrator's Guide*
- *Oracle Application Server Adapters for Files, FTP, Databases, and Enterprise Messaging User's Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction

With the growing need for business process optimization, efficient integration with existing back-end applications has become the key to success. To optimize business processes, you can integrate applications by using Oracle Application Server adapters. Adapters support a robust, light-weight, highly scalable, and standards-based integration framework, which enables disparate applications to communicate with each other. For example, adapters enable you to integrate packaged applications, legacy applications, databases, and Web services. Using adapters, you can ensure interoperability by integrating applications that are heterogeneous, provided by different vendors, based on different technologies, and run on different platforms.

> **Note:** This document addresses the integration of adapters with OC4J, Business Process Execution Language for Web Services (BPEL) Process Manager, OracleAS Integration InterConnect, and Oracle Enterprise Service Bus.

This chapter contains the following topics:

1.1 Features of Oracle Application Server Adapters

Oracle Application Server adapters provide the following benefits:

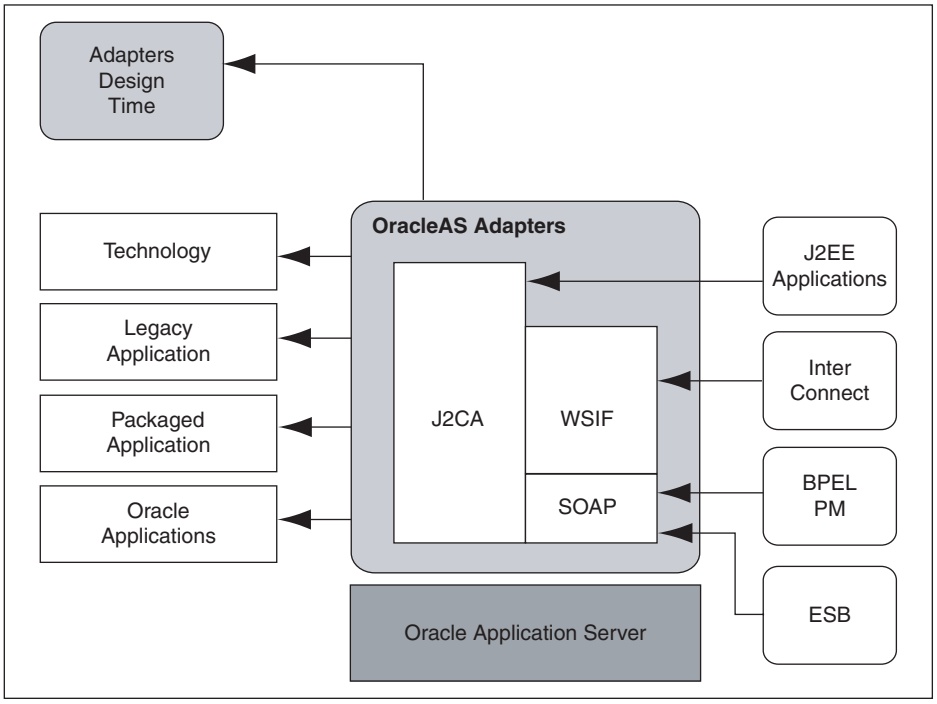
- Provide a connectivity platform for integrating complex business processes: Adapters integrate mainframe and legacy applications with enterprise resource planning (ERP), customer relationship management (CRM), databases, and messaging systems. Oracle provides more than 200 adapters to connect various packaged applications, such as SAP and Siebel, and databases. In addition, adapters integrate middleware messaging systems, such as MQSeries and Oracle Advanced Queuing, and legacy applications, such as Tuxedo, to provide a complete solution.
- Support open standards: Adapters are based on a set of standards such as J2EE Connector Architecture (J2CA), Extensible Markup Language (XML), Web service Invocation Framework (WSIF), Web service Inspection Language (WSIL), and Web service Definition Language (WSDL). The support for standards reduces the learning curve of a user and eliminates the dependency of users on a single vendor.

- Implement a Service-Oriented Architecture (SOA): The support for open standards enables adapters to implement an SOA, which facilitates loose coupling, flexibility, and extensibility.
- Use native APIs: Adapters support multiple ways of interfacing with the back-end system and provide various deployment options. Using native APIs, adapters communicate with the back-end application and also translate the native data to standard XML, which is provided to the client.
- Model data: Adapters convert native APIs to standard XML and back based on the adapter metadata configured during design time. Adapter configurations are defined during design time that which will be used by run-time components.
- Facilitate real-time and bidirectional connectivity: Adapters offer bidirectional communication with various back-end systems. This includes sending requests to back-end systems and receiving a response. Adapters also support the real-time event notification service. This service notifies about the back-end events associated with successful back-end transactions for creating, deleting, and updating back-end data. This two-way connectivity ensures faster, flexible, efficient integration, and reduces the cost of integration.
- Maximize availability: Adapters are based on J2CA 1.5 specification and deployed in the Oracle Containers for J2EE (OC4J), which is the J2CA container of Oracle Application Server. Adapters can, therefore, fully leverage the scalability and high availability of the underlying Oracle Application Server platform. In addition, adapters can be deployed on the JBoss and Weblogic platforms.
- Provide easy-to-use design-time tools: Adapters use design-time tools that provide a graphical user interface (GUI) to configure and administer adapters for fast implementation and deployment. In addition, the tools let you to browse, download, and configure back-end schemas.
- Support seamless integration with Oracle Application Server components: Adapters integrate with various Oracle Application Server components, such as Oracle Application Server Portal, and BPEL Process Manager, and J2EE applications, such as Enterprise Java Beans (EJBs), servlets, and Java applications.

1.2 Types of Oracle Application Server Adapters

There are three types of Oracle Application Server adapters: technology, packaged application, and legacy. Figure 1–1 illustrates the different types of adapters.

Figure 1–1 Types of Oracle Application Server Adapters



This section describes the following types of adapters:

- Technology Adapters
- Packaged-Application Adapters
- Legacy Adapters

1.2.1 Technology Adapters

Technology adapters integrate Oracle Application Server with transport protocols, data stores, and messaging middleware. These adapters include OracleAS Adapter for FTP, OracleAS Adapter for JMS, OracleAS Adapter for Database, OracleAS Adapter for Advanced Queuing, OracleAS Adapter for Files, and OracleAS Adapter for MQ Series. Technology adapters are currently available with the BPEL Process Manager installation.

1.2.2 Packaged-Application Adapters

Packaged-application adapters integrate Oracle Application Server with various packaged applications, such as SAP and Siebel. These adapters include OracleAS Adapter for Oracle Applications, OracleAS Adapter for PeopleSoft, OracleAS Adapter for SAP R/3, OracleAS Adapter for Siebel, and OracleAS Adapter for J.D. Edwards. Packaged-application adapters are available as part of the OracleAS Adapters CD.

1.2.3 Legacy Adapters

Legacy adapters integrate Oracle Application Server with legacy and mainframe applications. These adapters include OracleAS Adapter for Tuxedo, OracleAS Adapter for CICS, OracleAS Adapter for VSAM, OracleAS Adapter for IMS/TM, and OracleAS Adapter for IMS/DB. Legacy adapters are available as part of the OracleAS Adapters CD.

1.3 Types of Oracle Application Server Adapter Services

Adapters provide the following types of services to facilitate communication between applications:

- Request-Response (Outbound Interaction) Service
- Event Notification (Inbound Interaction) Service
- Metadata Service

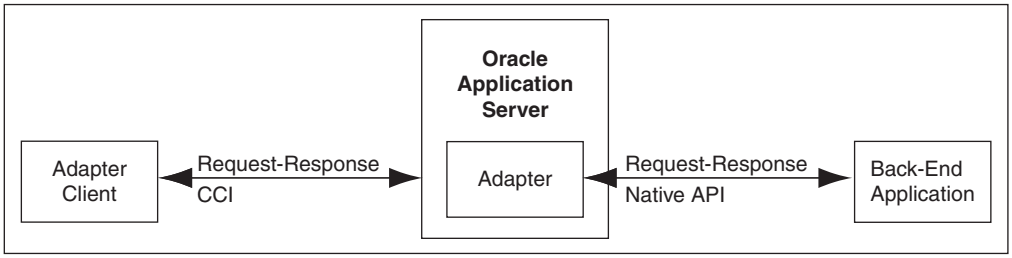
1.3.1 Request-Response (Outbound Interaction) Service

Adapters support the synchronous request-response service. The adapters receive requests from adapter clients, translate these requests into the native back-end data format, and call the appropriate method in the back-end application. In addition, the request-response service retrieves the back-end response to the Adapter Framework component after performing reverse translation. In J2CA terminology, this type of service is also known as outbound interaction.

The request-response service can be used to create, delete, update, and query back-end data as well as to call back-end workflows and transactions. For example, an OC4J application client can use OracleAS Adapter for SAP to create a customer within the SAP application.

Figure 1–2 illustrates the request-response service.

Figure 1–2 Request-Response Service



1.3.2 Event Notification (Inbound Interaction) Service

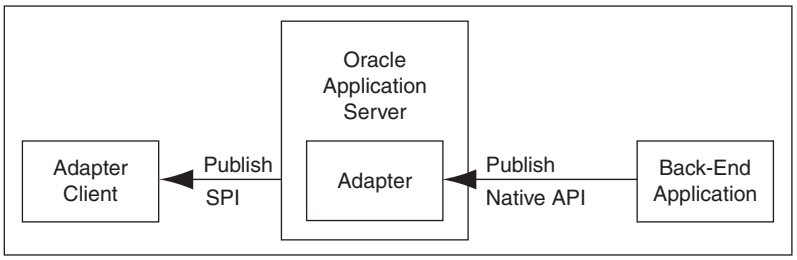
Adapters support the event-notification service, which is an asynchronous communication paradigm. In J2CA terminology, this type of service is also known as inbound interaction.

Adapters either listen or poll for back-end event changes. When listening for events, an adapter registers as a listener for the back-end application that is configured to push events to the adapter. The adapter can also poll the back-end application, which is usually a database or file, for the events required by the client application.

The event-notification service can be used to keep a track of back-end events associated with successful back-end transactions for creating, deleting, and updating back-end data.

Figure 1–3 illustrates the event-notification service.

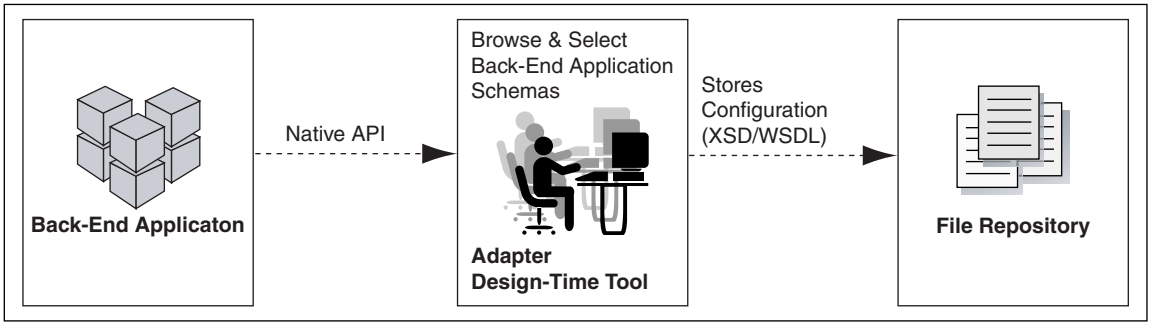
Figure 1–3 Event-Notification Service



1.3.3 Metadata Service

The adapter metadata definition stores information about the back-end connection and schemas for business objects and services. Adapters consist of a design-time component for browsing and storing metadata and a run-time component for running services. The adapter metadata definitions are generated as XML Schema Definition (XSD) and WSDL files. Figure 1–4 illustrates the metadata interaction.

Figure 1–4 Metadata Service



2

Technology Adapters

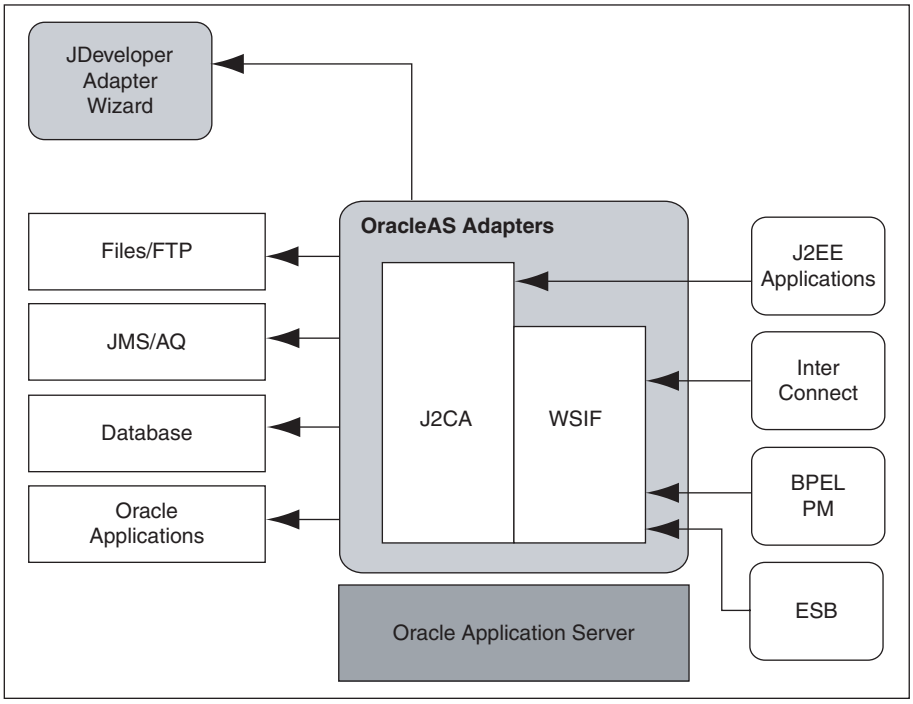
Technology adapters integrate Oracle Application Server with transport protocols, data stores, and messaging middleware. These adapters include OracleAS Adapter for FTP, OracleAS Adapter for JMS, OracleAS Adapter for Databases, OracleAS Adapter for Advanced Queuing, OracleAS Adapter for Files, and OracleAS Adapter for MQ Series.

This chapter contains the following topics:

2.1 Architecture

Technology adapters are based on J2EE Connector Architecture (J2CA) 1.5 standards and deployed as a resource adapter in the same Oracle Containers for J2EE (OC4J) container as BPEL Process Manager. OracleAS Adapter for Oracle Applications consists of the same architecture as the technology adapters. Figure 2–1 illustrates the architecture of technology adapters.

Figure 2–1 Technology Adapter Architecture



2.2 Design-Time Components

During design time, technology adapters use Oracle JDeveloper to generate the adapter metadata. The request-response service, also known as J2CA outbound interaction, and the event-notification service, also known as J2CA inbound interaction, are described in J2CA WSIF WSDL files. These WSDL files consist of J2CA extension elements. The J2CA WSDL elements are used by the Adapter framework to seamlessly integrate the J2CA 1.5 resource adapter with the BPEL Process Manager.

For more information about integration of technology adapters with BPEL Process Manager, refer to Section 5.2, "Adapter Integration with BPEL Process Manager".

Example 2–1 Generating WSDL Files for OracleAS Adapter for Databases

By using JDeveloper, you can configure OracleAS Adapter for Databases. This adapter helps you to perform data manipulation operations, call stored procedures or functions, and publish database events in real time. To configure adapter definitions, select **Database Adapter** from the Adapter Type dialog box, as shown in Figure 2–2.

Figure 2–2 Selecting the Database Adapter

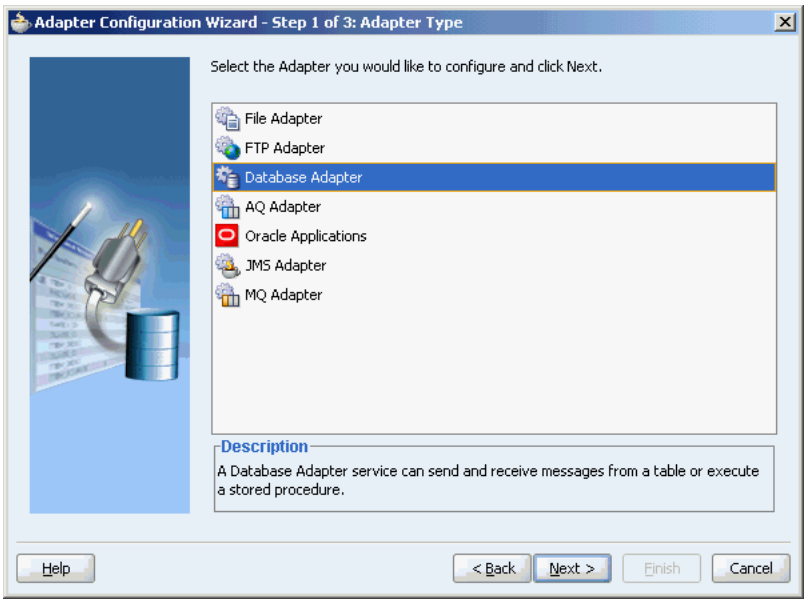


Figure 2–3 shows how to browse through the Import Tables window to select the required tables for the adapter.

Figure 2–3 Browsing for Required Tables

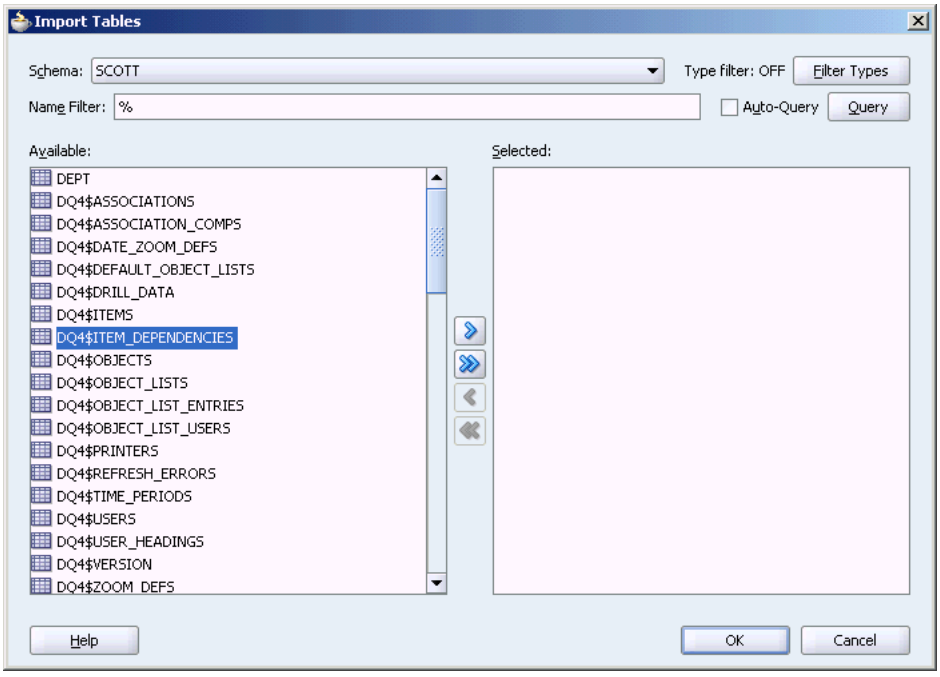
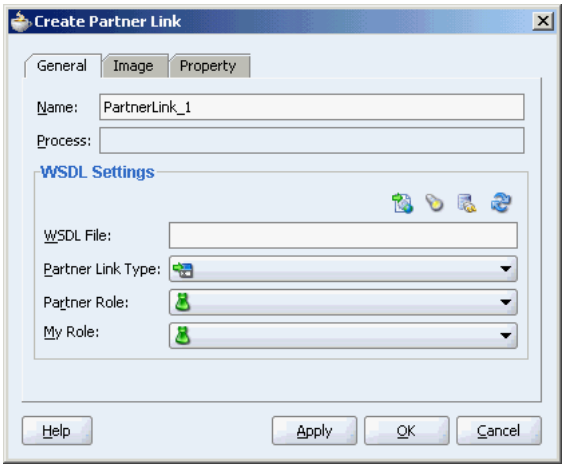


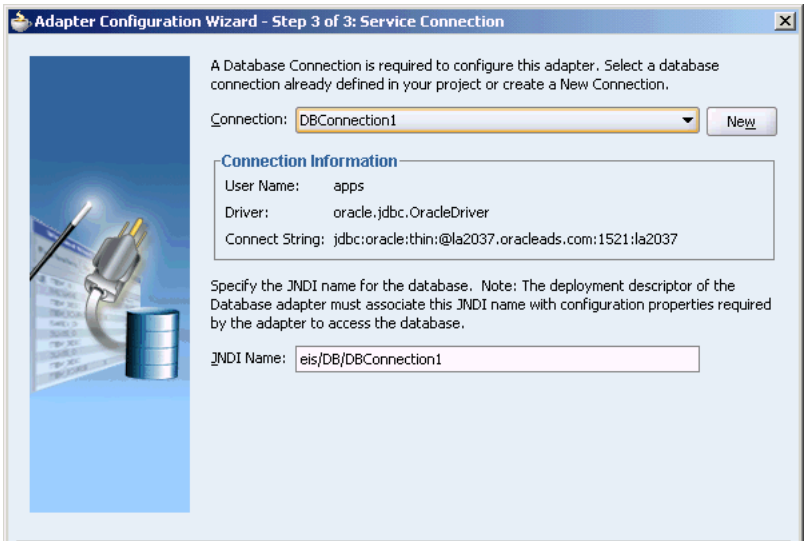
Figure 2–4 shows how to specify the WSDL settings for OracleAS Adapter for Databases.

Figure 2–4 Specifying WSDL Settings



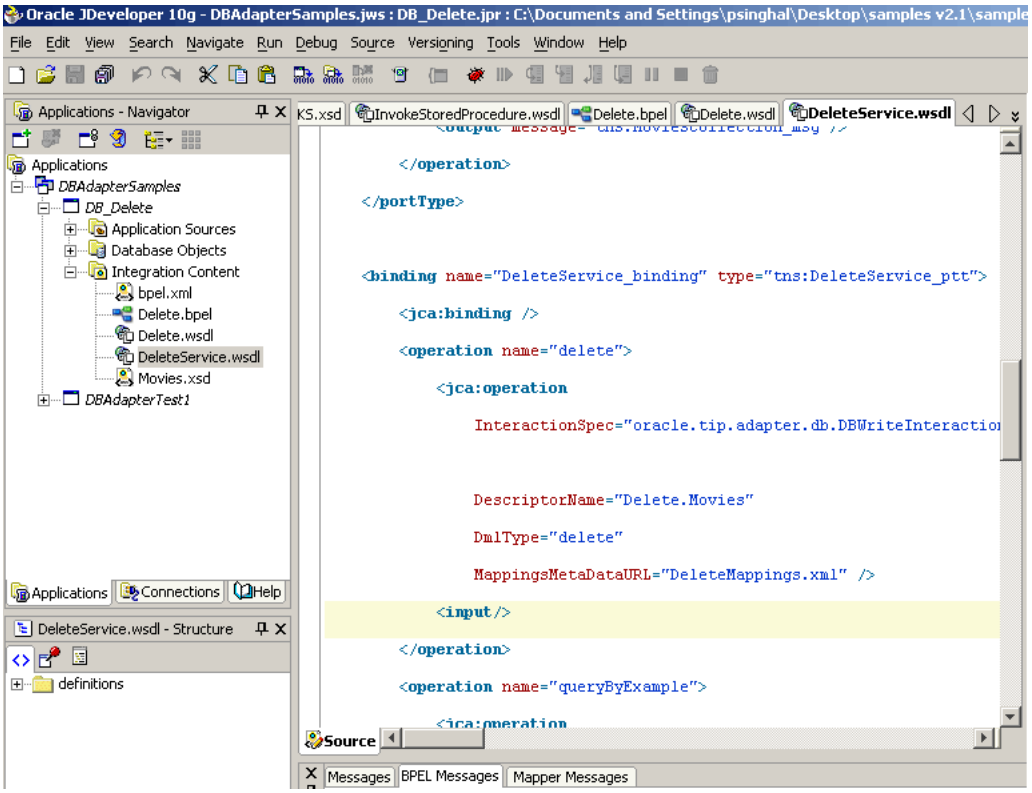
Next, you need to establish a database connection, select an operation type and select the required tables. The run-time connection parameters are specified in the `oc4j-ra.xml` file and linked to a Java Naming and Directory Interface (JNDI) name, which is specified during design time. Figure 2–5 shows the creation of a new database connection.

Figure 2–5 Creating a New Database Connection



Finally, JDeveloper generates a WSDL file with the J2CA binding for the OracleAS Adapter for Databases, as shown in Figure 2–6.

Figure 2–6 Structure of a WSDL File



Note: You can make technology adapters handle large XML payloads by including an additional setting in `bpel.xml`, under the activation agent as shown in the following example:

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

2.3 Run-Time Components

The run-time component of technology adapters is the J2CA 1.5 resource adapter for the specific back-end application. Technology adapters are deployed in OC4J, which is the J2CA container of Oracle Application Server. The BPEL Process Manager integrates with these J2CA 1.5 adapters through the Adapter Framework, which converts Web service messages into J2CA interactions and back.

BPEL Process Manager uses Adapter Framework to integrate the request-response service (J2CA outbound interaction) with a BPEL Invoke activity and publish the adapter events to a BPEL Receive activity. For more information about integration with the BPEL Process Manager, refer to Section 5.2, "Adapter Integration with BPEL Process Manager".

Note: Technology adapters can be integrated only with Business Process Execution Language for Web Services (BPEL) Process Manager.

2.4 Deployment

Technology adapters are deployed as J2CA 1.5 resource adapters within the same OC4J container as BPEL Process Manager during installation. Although technology adapters are physically deployed as J2CA 1.5 resource adapters, their logical deployment involves creating the Connection Factory entries for the J2CA 1.5 resource adapter by editing the `oc4j-ra.xml` file and using JDeveloper during design time. By using JDeveloper, you specify the JNDI name, which acts as a placeholder for the connection used when your service is deployed to the BPEL Server. This enables you to use different databases for development and later production. However, for the logical deployment changes to take effect, the OC4J container process should be restarted.

3

Packaged-Application Adapters

You can integrate Oracle Application Server with packaged applications, such as J.D. Edwards and Siebel by using packaged-application adapters. These adapters include OracleAS Adapter for Peoplesoft, OracleAS Adapter for SAP R/3, OracleAS Adapter for Siebel, OracleAS Adapter for Oracle Applications, and OracleAS Adapter for J.D. Edwards.

This chapter describes the architecture of packaged-application adapters. It contains the following topics:

3.1 Architecture

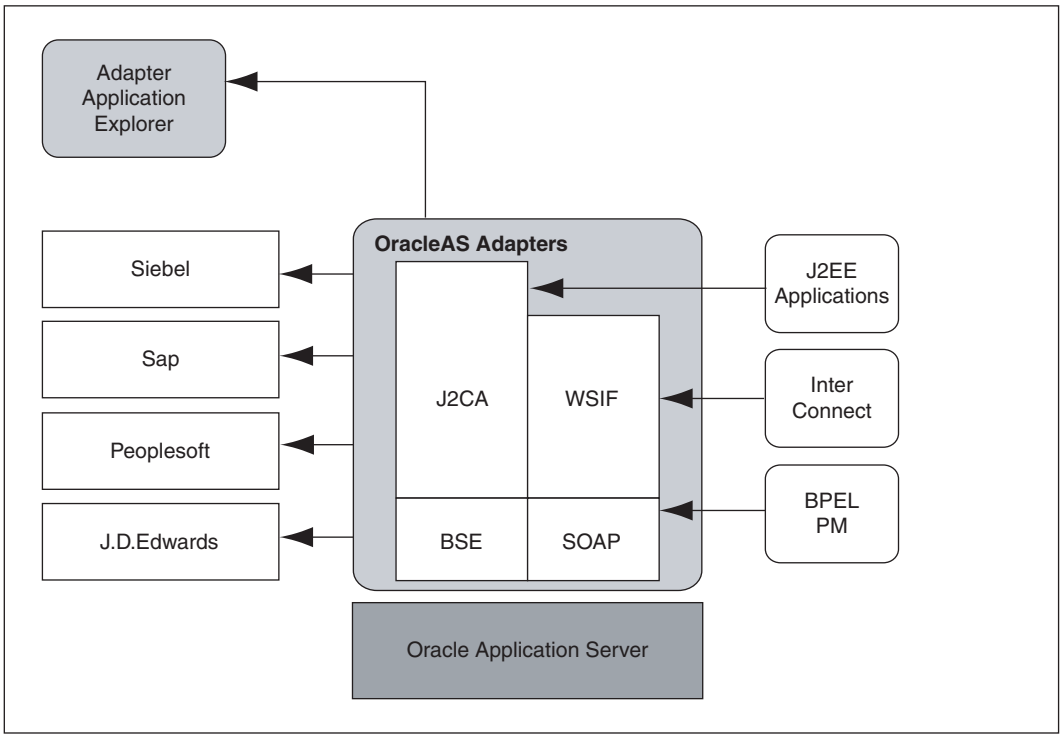
Packaged-application adapters can be deployed as J2EE Connector Architecture (J2CA) 1.5 resource adapters or as Web service servlets within the Oracle Containers for J2EE (OC4J) container. In addition to a J2CA interface, packaged-application adapters support the Web Service Definition Language (WSDL) and Simple Object Access Protocol (SOAP) interface. J2CA and Web service deployments of packaged-application adapters should have a repository project. In J2CA deployment, the resource adapter points to a repository project that can contain multiple back-end connection objects. The deployment descriptor, `oc4j-ra.xml`, points to the J2CA repository project and the connection name to access within the J2CA repository project. In the WSDL deployment, the WSDL repository project consists of a set of WSDL files that describe the adapter metadata.

> **Note:** Only four packaged-application adapters; OracleAS Adapter for SAP, OracleAS Adapter for Siebel, OracleAS Adapter for Peoplesoft, and OracleAS Adapter for J.D. Edwards, support WSDL and SOAP extensions in this release. The architecture of the OracleAS Adapter for Oracle Applications is similar to technology adapters.

The architecture of packaged-application adapters consists of OracleAS Adapter Application Explorer (Application Explorer), J2CA 1.5 resource adapter, and Business Services Engine (BSE).

Figure 3.1 illustrates the architecture of packaged-application adapters:

Figure 3–1 Packaged-Application Adapters Architecture



This section describes the following components of the packaged-application adapter architecture:

- Application Explorer
- BSE
- J2CA 1.5 Resource Adapter

3.1.1 Application Explorer

Application Explorer is a Java swing-based design-time tool for configuring packaged-application adapters. Using Application Explorer, you can configure the back-end application connection, browse the back-end application schemas, and expose these schemas as adapter services. Application Explorer is shipped with packaged application-specific plug-ins for browsing the back-end application-specific metadata.

You can use Application Explorer to create repository projects for either OracleAS Adapter J2CA or BSE. Each repository project can consist of multiple back-end application connections. The schemas are represented as either XML Schema Definition (XSD) for the OracleAS Adapter J2CA interface or as a WSDL with SOAP binding.

3.1.2 BSE

Application Explorer works in conjunction with the BSE, which is deployed in the Oracle Containers for J2EE (OC4J) container of the Oracle Application Server. BSE uses SOAP as a protocol for accepting requests from clients, interacting with the back-end application, and sending responses from the back-end application back to clients.

3.1.3 J2CA 1.5 Resource Adapter

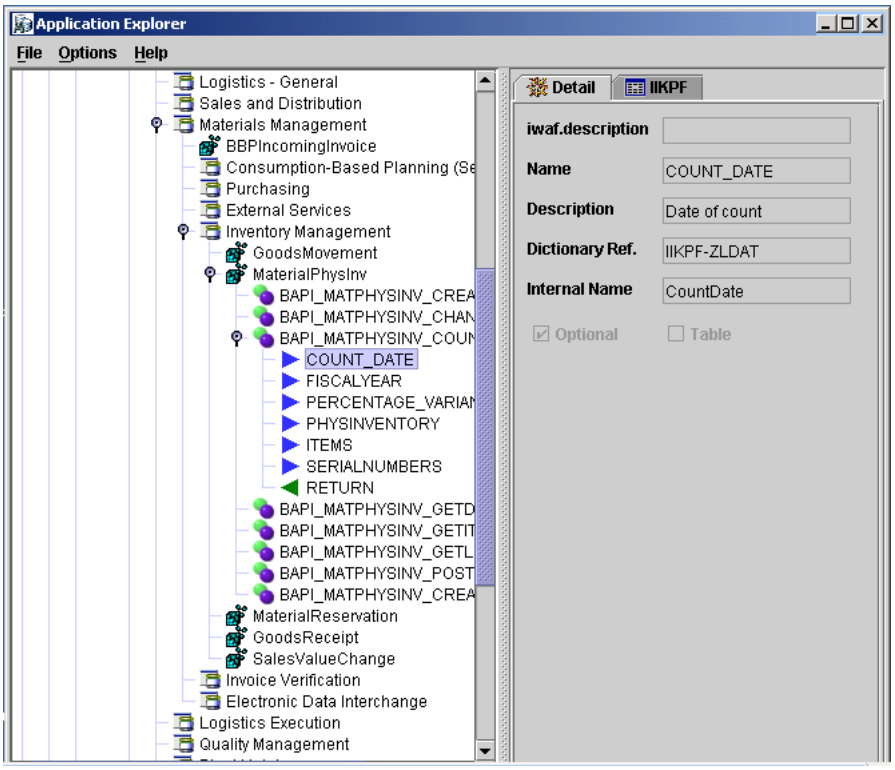
The J2CA 1.5 resource adapter consists of a Channel component for receiving back-end events.

3.2 Design-Time Components

Application Explorer is used to configure packaged-application adapters during design time. This tool is used to create a repository project for the J2CA 1.5 resource adapter, which contains a list of back-end connections. Application Explorer exposes back-end metadata as XSD and WSDL with J2CA extensions. The XSD metadata is used by OC4J application clients for integration through the J2CA Common Client Interface (CCI) Application Programming Interface (API). The WSDL with J2CA extension is used for integration with Business Process Execution Language for Web Services (BPEL) Process Manager. The BSE metadata can be defined as WSDL or SOAP.

Figure 3–2 shows the Application Explorer.

Figure 3–2 Application Explorer



Example 3–1 Generating XML Request Schema for OracleAS Adapter for SAP

You can use Application Explorer to establish a connection for OracleAS Adapter for SAP. For this, you must first define a target to OracleAS Adapter for SAP, as shown in the following figures:

Figure 3–3 Selecting OracleAS Adapter for SAP

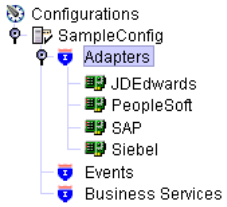
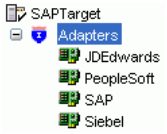
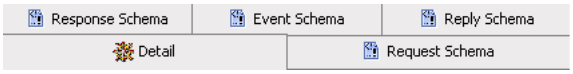


Figure 3–4 Defining a Target to OracleAS Adapter for SAP



After you have explored the SAP business function library and have selected an object, you can use Application Explorer to create the XML request schema and the XML response schema for that function. To view the XML for each schema type, select the required tab as shown in the following figure:

Figure 3–5 Viewing the XML Schema



3.3 Run-Time Components

The run-time components of packaged-application adapters include J2CA 1.5 resource adapter, BSE, and servlet. The OC4J application clients use the CCI API to directly interface with the J2CA 1.5 resource adapter. In this release, OC4J supports J2CA 1.0 specification only and the CCI API can be used for calling J2CA Outbound Interactions only. The J2CA 1.5 resource adapter integrates with BPEL Process Manager through the Adapter Framework. During run time, Adapter Framework translates the BPEL Process Manager service requests to J2CA calls and back based on the adapter metadata (WSDL or J2CA extension) configured during design time.

During run time, the WSDL files generated during design time are consumed by the integrating components. For example, the BPEL Process Manager uses Adapter Framework to integrate the request-response service (J2CA outbound interaction) with a BPEL invoke activity and to publish adapter events to a BPEL receive activity. For more information about integrating with BPEL PM, refer to Section 5.2, "Adapter Integration with BPEL Process Manager".

OC4J application clients, such as servlet, EJB, and Java applications, use the CCI API to interface with the J2CA Adapter. For more information about integration with OC4J, refer to Section 5.1, "Adapter Integration with OC4J".

3.4 Deployment

Packaged-application adapters are deployed as J2CA 1.5 resource adapters within the OC4J J2CA container during installation. The adapter needs to be in the same OC4J container as BPEL Process Manager for integration.

You can integrate any Web service client with the BSE servlet. The BSE exposes the underlying back-end functionality as Web services, which can be either WSDL or SOAP. BPEL Process Manager can integrate with the BSE layer as well through WSDL and SOAP binding.

BSE is deployed as a servlet within the OC4J container during installation. BSE can be remotely located and need not be in the same container as the BPEL Process Manager.

4

Legacy Adapters

Using legacy adapters, you can integrate Oracle Application Server with legacy and mainframe applications. These adapters include OracleAS Adapter for Tuxedo, OracleAS Adapter for CICS, OracleAS Adapter for VSAM, OracleAS Adapter for IMS/TM, and OracleAS Adapter for IMS/DB.

This chapter contains the following topics:

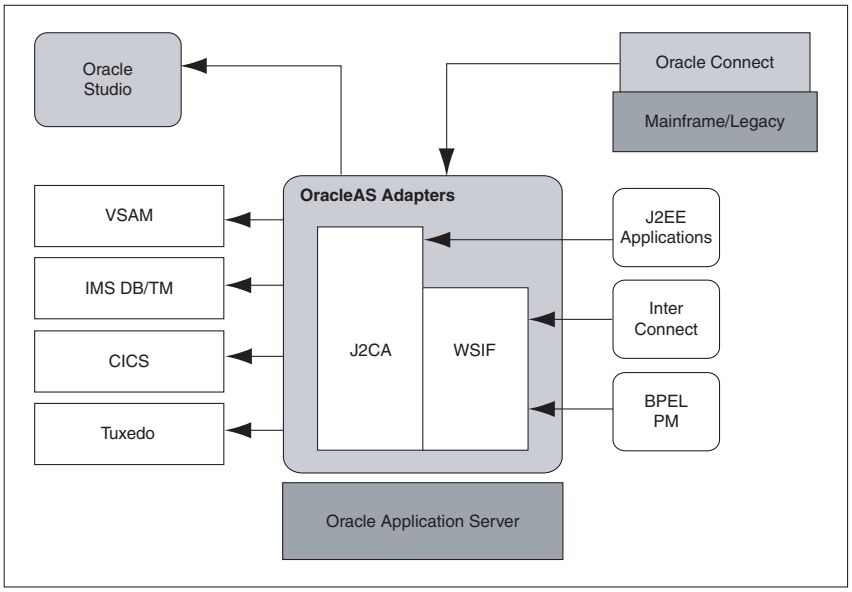
4.1 Architecture

Legacy adapters include the following components in the architecture:

- Oracle Connect
- Oracle Studio
- J2CA Adapter

Figure 4–1 illustrates the architecture of legacy adapters.

Figure 4–1 Legacy Adapter Architecture



4.1.1 Oracle Connect

Oracle Connect is a component that resides on the legacy and mainframe platform. It consists of native adapters for communicating with the mainframe application and data stores. Oracle Connect consists of the following components:

- Server Processes
- Native Adapters
- Daemon
- Repository

Server Processes

Oracle Connect consists of multiple servers to process client requests.

Native Adapters

Oracle Connect consists of various embedded native adapters to communicate with Tuxedo and IMS-TM transaction systems and database drivers to communicate with various databases and file systems on mainframe systems such as VSAM and IMS-DB. The native adapters convert application structures, such as the legacy COBOL applications data, to and from XML. The XSD schema is used for precise mapping between mainframe data and standard XML data.

Daemon

Daemon is an RPC-based listener that manages and maintains multiple server configurations. It runs on every machine running Oracle Connect and handles user authentication and authorization, connection allocation, and server process management.

When a client requests for a connection, the daemon allocates a server process to handle this connection. The allocated server process may be a new process or any process that might have been already running. Further communication between the client session and the server process is direct and does not involve the daemon.

However, the daemon is notified when the connection ends and the server process is either killed or being used by another client.

The daemon supports multiple server configurations called workspaces. Each workspace defines accessible data sources, applications, environment settings, security requirements, and server allocation rules. The daemon authenticates clients, authorizes requests for a server process within a certain server workspace, and provides clients with the required servers. The allocation of servers by the daemon is based on the workspace that the client uses. Thus, a client can access a data source using one workspace, where a server process is allocated from an existing pool of servers, or the client can access a data source using a different workspace, where a new server process is allocated for each client request. A fail-safe mechanism enables the specification of alternate daemons, which function as a standby for high availability.

Repository

Oracle Connect supports a repository for storing the XML-based schema and configuration information. There is a single repository for each Oracle Connect instance. The repository stores the following information:

- Oracle Connect configuration settings (including the Daemon settings to control client server communication)
- User profiles to enable single sign-on to multiple back-end applications and data sources
- Adapter metadata for each adapter, which includes adapter Request-Response and event services

4.1.2 Oracle Studio

Oracle Studio is the design-time tool for configuring the Oracle AS Adapters for mainframes. It enables you to configure the services, events, and connection information for native adapters. The configuration information is stored in the Oracle Connect repository on the legacy or mainframe application. In addition, it enables you to do management and monitoring of Oracle Connect. The Oracle Studio is available only on the Windows platform. The Oracle Studio is based on the IBM Eclipse GUI framework.

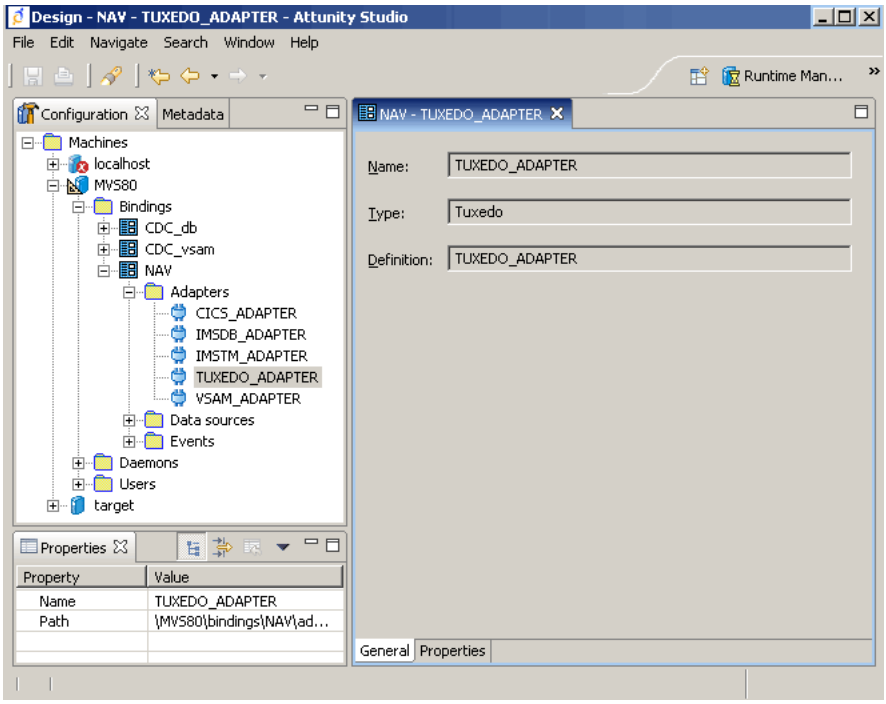
4.1.3 J2CA Adapter

The J2EE Connector Architecture (J2CA) adapter forwards the OC4J application client requests to the Oracle Connect application. Oracle Connect communicates with the mainframe application and forwards the response back to the J2CA adapter. The response might contain the transaction data or might contain the exception data if the request generated an error. The Business Process Execution Language for Web Services (BPEL) Process Manager and Oracle Application Server components integrate with Oracle Connect through the J2CA adapter.

4.2 Design-Time Components

To configure legacy adapters during design time use Oracle Studio, as shown in the following figure.

Figure 4–2 Oracle Studio



Example 4–1 Configuring OracleAS Adapter for Tuxedo

Using Oracle Studio, you can configure OracleAS adapter for Tuxedo, as shown in Figure 4–3.

Figure 4–3 Configuring OracleAS Adapter for Tuxedo

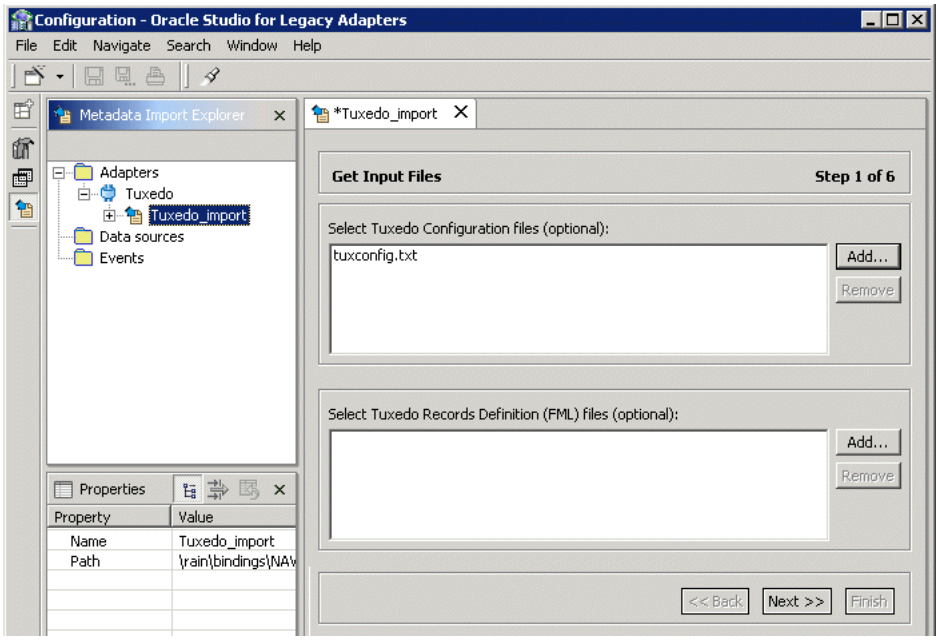
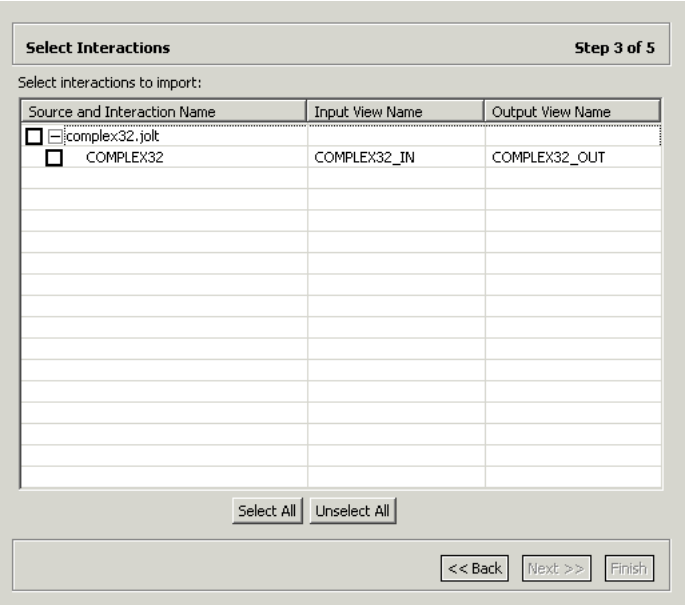


Figure 4–4 Selecting the Types of Interactions for OracleAS Adapter for Tuxedo



4.3 Run-Time Components

During run time, WSDL files generated during design time are consumed by the integrating components. For example, BPEL Process Manager uses Adapter Framework to integrate the request-response service (J2CA outbound interaction) with a BPEL invoke activity and to publish the events to a BPEL receive activity. For more information, refer to Section 5.2, "Adapter Integration with BPEL Process Manager"

The OC4J application client, such as servlet and EJB, uses the Common Client Interface (CCI) API to communicate with the J2CA Adapter. For more information, refer to Section 5.1, "Adapter Integration with OC4J".

4.4 Deployment

Legacy adapters are deployed as J2CA resource adapters within the OC4J J2CA container during installation. The adapter must be in the same OC4J container as that of the BPEL Process Manager for integration.

5

Adapter Integration with Oracle Application Server Components

Oracle Application Server adapters can be integrated with various components such as Oracle Containers for J2EE (OC4J), Oracle Application Server InterConnect, Business Process Execution Language for Web services (BPEL) Process Manager, and Oracle Enterpris Service Bus. This chapter discusses how to integrate adapters with OC4J, BPEL Process Manager, and OracleAS Integration InterConnect.

This chapter contains the following topics:

5.1 Adapter Integration with OC4J

Oracle Application Server adapters are deployed as J2EE Connector Architecture (J2CA) 1.0 resource adapters in an OC4J container. The resource adapter is used within the address space of the Oracle Application Server. This section provides an overview of OC4J and design-time and run time integration with an adapter. This section contains the following topics:

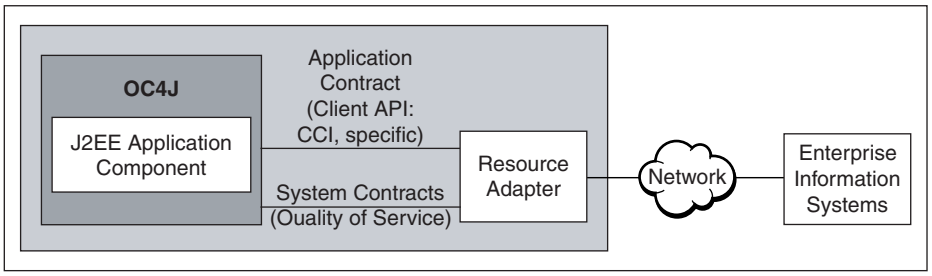
5.1.1 OC4J Overview

OC4J is the core J2EE run-time component of Oracle Application Server. OC4J is a J2EE 1.3 compliant container that runs on a standard JDK 1.4 Java Virtual Machine (JVM) and provides complete support for Java Server Page (JSP) files, servlets, enterprise java beans (EJBs), Web services, and all J2EE services. In addition, OC4J consists of a J2CA container for hosting J2CA resource adapters. J2CA defines standard Java interfaces for simplifying the integration of a J2EE server with various back-end applications.

All client applications run within the OC4J environment. To integrate an OC4J client application with a resource adapter, use the common client interface (CCI). The OC4J adapter clients include a servlet, EJB, or Java application client that implements the CCI Application Programming Interface (API). The CCI defines a standard client API for application components to access the back-end application.

On the other hand, the contract between the OC4J container and the resource adapter is defined by the service provider interface (SPI). Contracts define a standard between OC4J and adapters. The system handles these contracts automatically and hides them from the application developer. The following figure illustrates the CCI and SPI contracts:

Figure 5–1 Contracts between OC4J and Resource Adapter



The OC4J architecture includes the following set of system-level contracts:

- Connection management: Enables application components to connect to a back-end application and leverage any connection pooling support of the OC4J container. This leads to a scalable and efficient environment that can support a large number of components requiring access to a back-end application.
- Transaction management: Enables an application server to use a transaction manager to manage transactions across multiple resource managers. Most of the adapters support only local transactions (single-phase commit) and not XA transactions (two phase commit).
- Security management: Provides authentication, authorization, and secure communication between the J2EE server and the back-end application. The OC4J supports both container-managed (the Oracle Application Server is responsible for sending security credentials to the back-end application) and component-managed sign-on (the J2CA adapter is responsible for forwarding the security credentials to the back-end application).

5.1.2 OC4J Integration with Adapters

Oracle Application Server adapters are J2CA 1.5 based and deployed within OC4J during installation. The OC4J component supports only the J2CA 1.0 specification in this release. The J2CA resource adapter is packaged into a Resource Adapter Archive (RAR) file using the Java Archive (JAR) format. An RAR file contains a correctly formatted deployment descriptor `(/META-INF/ra.xml)`. In addition, it contains declarative information about the contract between the OC4J and resource adapter.

OC4J generates the corresponding `oc4j-ra.xml` file during the deployment of the J2CA adapter. The `oc4j-ra.xml` file is the deployment descriptor for a resource adapter. It contains deployment configurations for deploying resource adapters to OC4J, which includes the back-end application connection information as specified in the deployment descriptor of the resource adapter, Java Naming and Directory Interface (JNDI) name to be used, connection pooling parameters, and resource principal mapping mechanism and configurations.

Design Time

Use the adapter design-time tool to generate XML Schema Definition (XSD) files for the adapter request-response service. The OC4J clients use these XSD files during run time for calling the J2CA outbound interaction. Packaged-application adapters use

OracleAS Adapter Application Explorer (Application Explorer), Legacy adapters use OracleAS Studio, and technology adapters use JDeveloper.

Run Time

Oracle Application Server adapters are based on the J2CA 1.5 specification but are deployed as the J2CA 1.0 resource adapter within the OC4J container in this release. The J2CA 1.0 specification addresses the request-response service also known as outbound interaction and does not address the asynchronous publication of back-end events by the adapter. The J2CA 1.5 specification addresses the life-cycle management, message-inflow (for Adapter Event publish) and work management contracts.

5.2 Adapter Integration with BPEL Process Manager

Oracle Application Server adapters can be integrated with BPEL Process Manager. This section discusses the follows topics:

- BPEL Process Manager Overview
- BPEL Process Manager Integration with Adapters

5.2.1 BPEL Process Manager Overview

BPEL Process Manager is a comprehensive solution for creating, deploying, and managing BPEL business processes. BPEL Process Manager is based on the Service Oriented Architecture (SOA) to provide flexibility, interoperability, reusability, extensibility, and rapid implementation. BPEL Process Manager reduces the overall cost of management, modification, extension, and redeployment of existing business processes. Each business activity is a self-contained, self-describing, modular application with an interface that is defined by a WSDL file and the business process that is modeled as a Web service.

The following section discusses how Technology adapters gather and publish statistics for every message they process, either inbound or outbound in BPEL Process Manager 10.1.3:

5.2.1.1 Adapter Statistics

In BPEL Process Manager 10.1.3 several Technology adapters, such as File, JMS and Database, gather and publish statistics for every message they process either inbound or outbound. The statistics are broken down into categories and individual tasks. The following is an example of how statistics are broken down in an outbound process:

- Adapter Preprocessing
  - Preparing `InteractionSpec`
- Adapter Processing
  - Setting up Callable Statement
  - Invoking Database
  - Parsing Result
- Adapter Postprocessing

The adapter statistics can be viewed in the BPEL Console, from the same page where the BPEL engine statistics are available. The following are the steps to view the adapter statics in the BPEL Console:

1. From the front BPEL PM Console page, follow the link `Manage BPEL Domain`.

2. On the next page, click the link `Adapter Statistics`.

   This shows a list of all currently active inbound and outbound adapter interactions, and the average execution time for the various steps each adapter performs.

   To reset the statistics gathering, click the link `Clear Statistics`.

5.2.2 BPEL Process Manager Integration with Adapters

Adapter Framework is used for the bidirectional integration of the J2CA 1.5 resource adapters with BPEL Process Manager. Adapter Framework is based on standards and employs the Web service Invocation Framework (WSIF) technology for exposing the underlying J2CA interactions as Web services.

Design Time

While integrating adapters with BPEL Process Manager, the underlying adapter services are exposed as WSDL files with the J2CA extension. The following table lists the design time tools used for generating WSDL files for various types of adapters.

Adapter	Tool
Technology and OracleAS Adapter for Oracle Applications	JDeveloper
Packaged application	Application Explorer
Legacy	Oracle Studio

WSDL files are created for both Request-Response and Event-Notification services of an adapter. The J2CA extension contains J2CA elements that are required by Adapter Framework during run time to convert Web service messages to J2CA Interactions and back. The J2CA WSDL extension elements contain the metadata for Adapter Framework to call any Request-Response service and activate any inbound J2CA 1.5 endpoint to receive inbound events. The J2CA extension elements for the request-response service contains the JNDI location and `InteractionSpec` details for calling an outbound interaction. The J2CA extension elements for the event-notification service contains the resource adapter class name and `ActivationSpec` parameters for publishing an adapter event through the J2CA inbound interaction.

Figure 5–2 illustrates the design time tool, JDeveloper, used by technology adapters.

Figure 5–2 Design Time Configuration of Technology Adapters

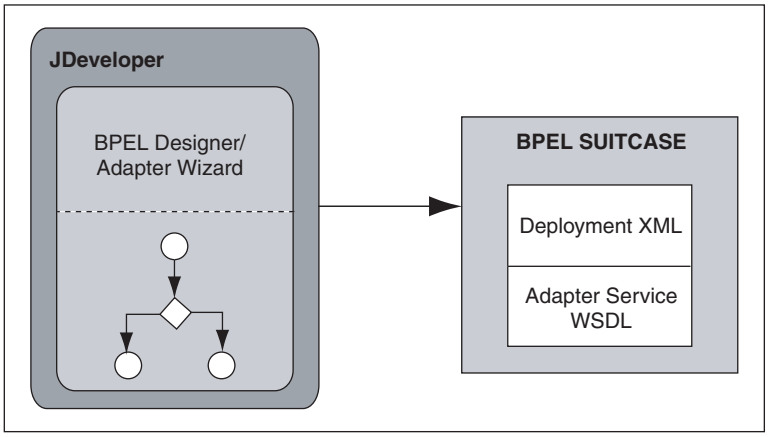
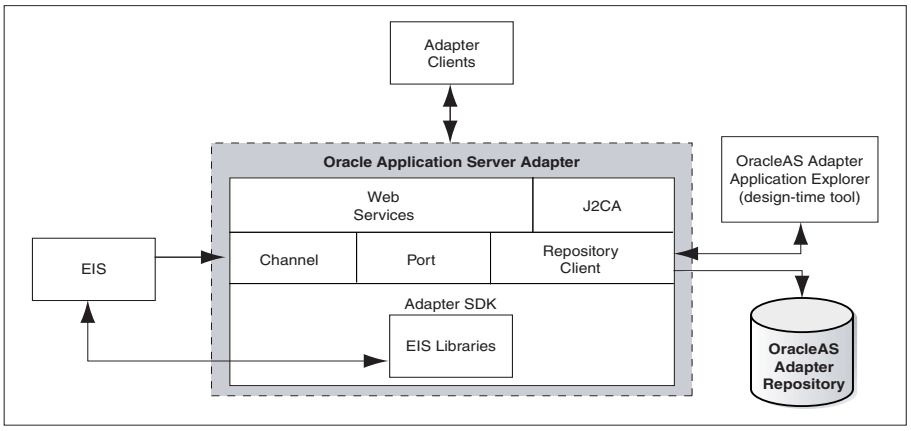


Figure 5–3 illustrates the design-time tool for configuring packaged-application and legacy adapters. In this figure, the design-time tools are used to expose adapter metadata as WSDL files. The WSDL files are consumed by BPEL Process Manager during run time.

Figure 5–3 Configuring Legacy and Packaged-Application Adapters



Run Time

Oracle Application Server adapters are based on J2CA 1.5 specification and deployed with the Oracle BPEL Process Manager in the same OC4J container. Adapter Framework acts as the glue layer that integrates the standard J2CA 1.5 resource adapter with the Oracle BPEL Process Manager during run time. Adapter Framework acts as a psuedo-J2CA 1.5 container, which enables implementation although the Oracle Application Server OC4J container supports only J2CA 1.0.

Adapter Framework consists of a WSIF J2CA Provider for wrapping the J2CA interactions as Web services. In addition, Adapter Framework translates Web service messages to J2CA interaction message based on the WSDL files generated during design time.

The WSIF Provider converts the Web service invocation (from BPEL PM Invoke activity) to a J2CA outbound interaction call and performs the reverse conversion in the other direction. In other words, Web service invocation launched by the BPEL Invoke activity is converted to a J2CA CCI outbound interaction and the J2CA response is converted back to a Web service response. This end-to-end invocation is

synchronous. The WSIF Provider also supports the one-way asynchronous J2CA outbound interaction invocation. The WSIF Provider element hides the J2CA implementation details from the BPEL Process Manager process and the Web service details from the J2CA 1.5 resource adapter.

BPEL Process Manager Integration with Outbound Interaction

BPEL Process Manager uses the WSIF technology to start the request-response service of the resource adapter. WSIF is used for calling Web services and separating the abstract part of the Web service definition from the physical binding (transport and protocol) to generate a Service-Oriented Architecture (SOA). The following list summarizes the process of BPEL Process Manager integration with the outbound interaction.

- During design time, adapter services are exposed as WSDL files and consumed during configuration of the `PartnerLink` activity of the BPEL process.
- The WSDL extensions specify the JNDI address of the resource adapter, `InteractionSpec` class name, `InteractionSpec` parameters.
- During run time, the Invoke activity of the BPEL Process Manager is used to call the `PartnerLink` activity, which is J2CA Resource Adapter outbound interaction.
- The WSIF J2CA provider receives the Web service request from BPEL Process Manager.
- The WSDL file associated with this Web service request contains the JNDI name of the J2CA `ConnectionFactory` class in the WSDL extension element.
- The WSIF provider performs the JNDI lookup and obtains a Connection instance.
- The WSIF provider creates an `InteractionSpec` Bean object from the WSDL extension element jca:operation.
- The WSIF provider starts the J2CA outbound interaction by passing the `InteractionSpec`, the `InputRecord`, and an empty `OutputRecord` parameter.
- The J2CA 1.5 resource adapter calls the back-end application and returns the back-end application response to the WSIF Provider.
- This XML record is translated to a Web service Response for consumption by the BPEL PM instance.
- The whole end-to-end scenario is synchronous and any J2CA exception is forwarded as a Web service fault message.

BPEL Process Manager Integration with Inbound Interaction

BPEL Process Manager receives events from the J2CA 1.5 resource adapter through Adapter Framework, which is the psuedo J2CA 1.5 container and implements the message-inflow contracts for receiving events from the adapter. The J2CA inbound interaction is captured in a WSDL file during design time. The J2CA Inbound WSDL binding section contains the J2CA 1.5 `ActivationSpec` parameter. The `ActivationSpec` parameter captures the inbound connectivity and inbound interaction details (according to J2CA 1.5 specification). The J2CA Inbound WSDL Service section contains the J2CA 1.5 `ResourceAdapter` class name. In addition, the Service section can optionally contain a JNDI location. The following list summarizes the process of BPEL Process Manager integration with the inbound interaction:

- The `ResourceAdapter` class name and the `ActivationSpec` parameter are captured in the WSDL extension section of the J2CA inbound interaction WSDL

during design time and made available to BPEL Process Manager and Adapter Framework during run time.

- An instance of the J2CA 1.5 `ResourceAdapter` class is created and the `Start` method of the J2CA `ResourceAdapter` class is called.
- Each inbound interaction operation referenced by the BPEL Process Manager processes results in invoking the `EndPointActivation` method of the J2CA 1.5 `ResourceAdapter` instance. Adapter Framework creates the `ActivationSpec` class (Java Bean) based on the `ActivationSpec` details present in the WSDL extension section of the J2CA inbound interaction and activates the endpoint of the J2CA 1.5 resource adapter.
- The Adapter Framework `MessageEndpoint` implementation implements the `javax.resource.cci.MessageListener` interface. The J2CA 1.5 resource adapter calls the `onMessage()` method in this `MessageEndpoint` when it receives a back-end application event. The J2CA 1.5 resource adapter creates an instance of the `MessageEndpoint` implementation through the `MessageEndpointFactory` provided to the resource adapter during `endpointActivation`.
- Adapter Framework receives the event through the `MessageListener` class and forwards it to the Receive activity of the BPEL Process Manager instance.
- When the BPEL process is stopped, all associated inbound end points are deactivated through the `endPointDeactivation` method implemented by the resource adapter.

BPEL WSIF JCA Connection Pool

In the case of J2CA adapters, particularly the JDBC based ones, such as DB adapters and AQ adapters there are two kinds of connection management at play: one for inbound (endpoint) activations (BPEL Receive) and one for outbound interactions (BPEL Invoke).

In the case of inbound activations, the J2CA adapter is fully in-charge of connection creation and recovery. The Adapter Framework (AF) can only be requested to lookup and provide a J2CA `ConnectionFactory` handle to the adapter through its `ActivationSpec`. This is possible only if it implements a certain interface, which it can use to create connections, thereby going through the Application Server connection manager. Whenever a managed (JDBC) connection goes bad, the adapter must close the J2CA connection handle (and subsequently the managed connection if `destroy()` is called by the Application Server), enter a temporary recovery loop and then try to re-establish a new connection.

In the case of outbound interactions (WSIFor J2CA) each WSIF port caches tuples of the following:

- ConnectionFactory
- ConnectionSpec
- Connection
- Interaction
- InteractionSpec

As the BPEL engine typically invokes the WSIF port concurrently with any number of threads, the size of the cache will typically reflect the highest concurrency level at any given time. The cache can be tuned to auto-expire unused tuples after a configured idle period (interactions and connection handles are then closed).The cache greatly

improves performance in high load environments, for example, Retek (8 million transactions every hour).

If just one JCA adapter interaction using the cache throws a `ResourceException`, all members of the cache are closed and released immediately (purged) so new interactions will have to re-create (fresh) members to the cache. The BPEL engine has a feature known as *PartnerLink* retry which can be configured for each invoke. Thus, any JCA adapter invoke or interaction which throws a `ResourceException` marked as `Retryable`, will make the engine retry the Invoke (DB update) which will then repopulate the WSIF port cache (if the DB has become available again: typically immediately the case with RAC).

For non-transactional adapters (`adapterMetadata.supportsLocalTransactionDemarcation() == false`), such as File adapter, the J2CA connection cache only contains one member. Thus all threads coming through will multiplex over the same CCI Connection handle.

The JCA connection cache can be enabled or configured explicitly using the following bpel.xml partnerlink properties:

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

Normally this property is derived from the declared transactional support of the adapter. For example, the File adapter does not use this connection pool because it is multi-thread safe, but that can be overridden through the following property:

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

If the property mentioned in the preceding example is not specified, then the size of the connection pool is assumed to be unbounded. This applies on a per `WSIFPort` (partnerlink) basis.

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

Maximum age of idle connections in the pool is important because some type of connections hold on to expensive external resources, for example DB shadow processes which is measured in ms, as shown in the following example:

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

Finally, the property mentioned in the preceding example determines how frequently the connection pool should be scanned for idle connections, also measured in ms.

5.3 Adapter Integration with OracleAS Integration InterConnect

Packaged-application adapters including OracleAS Adapter for PeopleSoft, OracleAS Adapter for SAP R/3, OracleAS Adapter for Siebel, and OracleAS Adapter for J.D. Edwards can be integrated with OracleAS Integration InterConnect. In addition to J2CA deployment, packaged-application adapters support web service servlet deployment, which is called Business Service Engine (BSE). This integration can be deployed within an enterprise or across enterprise boundaries through the Internet. OracleAS Integration InterConnect is designed as a hub and spoke system. In this system, spokes function as OracleAS Integration InterConnect adapters that access other applications and systems. The hub acts as a OracleAS Integration InterConnect repository server, which is a standalone Java application. The OracleAS Integration InterConnect repository is a database that stores the design-time metadata definition.

Deployment and Integration through BSE

BSE integrates with OracleAS Integration InterConnect products and enables access to packaged applications. It helps businesses to integrate heterogeneous information systems faster, more cost effectively, and more uniformly by implementing standards that provide simple and consistent connections between these systems over the Internet. Business functions and data can be exposed as a Web Service, and tools on other platforms can use these services with minimal effort. The OracleAS Adapter Business Services Engine supports the following standards:

- A Simple Object Access Protocol (SOAP) request makes a Web Service perform a task.
- Web Services Definition Language (WSDL) describes all of the SOAP calls that can be used for a set of services.
- A Universal Description, Discovery, and Integration (UDDI) server acts as a directory of relevant Web Services and the WSDL that describes them.

No programming knowledge is necessary to generate these services, which reuse functionality from ERP or CRM systems, legacy applications, and diverse data sources.

BSE is deployed as a servlet within the OC4J container of the Oracle Application Server, and it depends on the OC4J container for scalability and high availability. The Swing-based design-time tool, Application Explorer, is used for browsing the EIS metadata that it passes to BSE. Adapter creates the WSDLs for the various adapter services and stores the WSDL schemas in an XML-based file repository or an Oracle Database repository. BSE must be running during both design time and at runtime.

Integration with OracleAS Integration InterConnect

The OracleAS Integration InterConnect EIS Adapter Plugin integrates the OracleAS Web Services adapter with OracleAS Integration InterConnect by translating XML payloads between XSD and DTD formats, respectively.

The OracleAS Integration InterConnect product supports four messaging paradigms: implemented procedures (request-response service), subscribe (one-way request service), publish (event service), and invoked procedure (OracleAS Integration InterConnect is the server in this case and EIS is the client making the request).

OracleAS Integration InterConnect EIS Adapter Plugin supports only the first three messaging paradigms. It uses SOAP for implemented procedure and subscription and RMI for the request-response and event-notification services.

The following components are used in integrating with OracleAS Integration InterConnect:

- BSE, which is the OracleAS Adapter deployed as a Web Services servlet within OC4J
- OracleAS Integration InterConnect EIS Adapter Plugin to talk to the BSE
- OracleAS Integration InterConnect deployment
- Oracle iStudio
- Application Explorer

OracleAS Integration InterConnect at Design Time

During design time, Application Explorer configures BSE. In addition to the standard WSDL schemas, BSE generates DTD schemas and stores them locally in a file system specified by the user. The OracleAS Integration InterConnect iStudio tool includes a DTD browser plugin, and can be used to convert DTD schemas to the native OracleAS

Integration InterConnect schema and store these in the OracleAS Integration InterConnect repository. More specifically, the following steps complete configuration for Request-Response and Event-Notification scenarios during the design time.

Request-Response; Implemented Procedures and Subscribe

1. Configure the BSE for request-response service using Application Explorer. The WSDL schemas are stored in the OracleAS Adapter repository and the DTD schemas are saved to a local file system.
2. Use OracleAS Integration InterConnect iStudio DTD browser to consume the DTD schemas generated in the previous step and store the configuration information in the OracleAS Integration InterConnect repository.

Event-Notification; Publish

1. Configure the BSE for event services, including the RMI port definition, using Application Explorer. The WSDL schemas are stored in the Oracle Adapter repository and the DTD schemas are saved to a local file system.
2. Use OracleAS Integration InterConnect iStudio DTD browser to consume the DTD schemas generated in the previous step and store the configuration information in the OracleAS Integration InterConnect repository.

OracleAS Integration InterConnect During Runtime

The OracleAS Integration InterConnect EIS Adapter Plugin loads the metadata from the OracleAS Integration InterConnect repository using its own API. The OracleAS Integration InterConnect EIS Adapter Plugin interfaces with the EIS application and translates the data between EIS native format and the Application View, a format recognized by OracleAS Integration InterConnect. This interaction is illustrated in Figure 6. More specifically, the following steps describe run-time behavior for Request-Response and Event-Notification scenarios.

Request-Response; Implemented Procedures and Subscribe

The OracleAS Integration InterConnect hub sends the request to OracleAS Integration InterConnect EIS Adapter Plugin, which creates a SOAP client request and makes a call to the BSE. The adapter invokes the EIS and forwards the EIS response as a SOAP response back to the OracleAS Integration InterConnect EIS Adapter Plugin, which in its turn translates the SOAP response to appropriate OracleAS Integration InterConnect format and forwards it on to the OracleAS Integration InterConnect hub.

Event-Notification; Publish

BSE receives an EIS event-notification through the Channel component. It then forwards this event as an RMI request to the OracleAS Integration InterConnect EIS Adapter Plugin, which acts as an RMI server and consumes this event, translates it to the appropriate OracleAS Integration InterConnect format and forwards it to the OracleAS Integration InterConnect hub.

Deployment and Integration through J2CA

The OracleAS Adapter J2CA supports only synchronous outbound interactions, such as the request-response service, not event-notification. At runtime, an EJB, a servlet or a Java application client communicates with the OracleAS Adapter J2CA through the CCI API. The OracleAS Adapter J2CA is deployed within OC4J as a standard J2CA 1.0 Resource Adapter, and runs in managed mode within the OC4J container. Similar to the BSE, it is configured using Application Explorer. The OracleAS Adapter J2CA is in practice a J2CA wrapper around the adapter SDK.

An OracleAS Adapter J2CA is universal; its repository project can contain more than one EIS connection and it can talk to more than one EIS type. The repository project can be in a file or database repository; its `ra.xml` file contains the repository connection parameters and the repository project name. Multiple managed ConnectionFactory objects can be specified in the `oc4j-ra.xml` file generated by OC4J, each pointing to a different repository project and JNDI name. To connect to a particular EIS, the J2CA CCI ConnectionSpec class is used to specify the EIS type and the EIS connection name.

The basic outline for using CCI with the OracleAS Adapter J2CA consists of the following steps:

1. Find a `ConnectionFactory` object for the OracleAS Adapter J2CA.
2. Create a `ConnectionSpec` object that uses the EIS type and EIS connection name.
3. Create a `Connection` to the EIS using the `ConnectionFactory` and `ConnectionSpec` objects.
4. Create an `Interaction` object based on the request and response XSD schemas.
5. Call the `execute()` method on the Interaction.
6. After the required interactions have been processed, close the `Interaction` and `Connection` objects.

5.4 Adapter Integration with Oracle Enterprise Service Bus

This section comprises the following topics:

- Oracle Enterprise Service Bus Overview
- Oracle Enterprise Service Bus Integration with Adapters

5.4.1 Oracle Enterprise Service Bus Overview

An Enterprise Service Bus (ESB) moves data among multiple endpoints: both within and outside of an enterprise. It uses open standards to connect, transform, and route business documents (as Extensible Markup Language (XML) messages) among diverse applications. It enables monitoring and management of business data, without having much impact on existing applications. An enterprise service bus is the underlying infrastructure for delivering a service-oriented architecture (SOA) and event-driven architecture (EDA).

Oracle Enterprise Service Bus is the base for services using SOA and EDA. At its core, it is a loosely coupled application framework that provides your business with increased flexibility, reusability, and overall responsiveness in a distributed, heterogeneous, message-oriented environment using industry standards.

Connectivity is provided through adapter services and Simple Object Access Protocol (SOAP) invocation services.

5.4.2 Oracle Enterprise Service Bus Integration with Adapters

The services you create with Oracle JDeveloper ESB Designer enable you to integrate the Oracle Enterprise Service Bus with file systems, database tables, database queues, Java Message Services (JMS), and Oracle E-Business Suite and any SOAP service.

Services are the main part of the enterprise service bus. You design an Oracle Enterprise Service Bus by creating a variety of services, to move messages onto, across, and off of the service bus.

Moving Data on to the Oracle Enterprise Service Bus

To move data on to the service bus, you use inbound adapter services or have an external application call an ESB service.

Moving Data off the Oracle Enterprise Service Bus

To move data off of the service bus, you use an outbound adapter service or invoke an external SOAP service.

Moving Data Across the Oracle Enterprise Service Bus

To move data across the service bus and transform the data structure from the structure presented by the source application to the structure required by the target application, you use routing services.

Oracle Enterprise Service Bus provides support for creating services for the Oracle Technology adapters. The Oracle Technology adapters enable you to integrate mainframe and legacy applications with enterprise resource planning (ERP), customer relationship management (CRM), database, and messaging systems.

5.4.2.1 Introduction to Adapter Services

Oracle JDeveloper ESB Designer provides wizards that assist you in creating inbound and outbound adapter services. The wizard collects the necessary information to generate the WSDL file that defines the service.

Adapters services can be configured as inbound or outbound adapters services. Inbound adapter services send messages to the Oracle Enterprise Service Bus, while outbound adapter services send messages to an application or system external to the Oracle Enterprise Service Bus.

5.4.2.2 Adapter Framework Features Supported/ Not Supported by Oracle Enterprise Service Bus

The section describes various Adapter Framework features that are supported and features that are not supported by the Oracle Enterprise Service Bus in the 10.1.3 release.

Adapter Framework Features Supported by Oracle Enterprise Service Bus

The following features are supported by the Oracle Enterprise Service Bus in the 10.1.3 release:

Partnerlink / Activation Agent (bpel.xml) Properties

In 10.1.3, the adapter framework receives partnerlink- and/or activation agent properties through a `java.util.Map` property, handed to the `ESBActivationAgent` through

```
oracle.tip.esb.jca.ESBActivationAgent.activateEvent
```

The origin of this map is `bpel.xm`, available within BPEL (provided by the BPEL Activation Framework), but can also be sourced from any type of name or value set. These properties are eventually provided to a `ResourceAdapter` through the `EndpointPropertiesAssociation` interface.

Outbound WSIF `execute()` retry

The WSIF operation, `executeRequestResponseOperation()` method is invoked in the method,

oracle.tip.esb.serviceimpl.outadapter.OutboundAdapterService.nextService().

In 10.1.3, any WSIFException is being linked to an EsbServiceException, and then propagated out of processBusinessEvent.

The WSIFException has a boolean flag named isRetryable(), which can be used to determine whether a service call can be retried or should be aborted.

The retry regiment could be controlled by the same partnerlink properties currently used by BPEL. For example, retryInterval (in seconds), retryMaxCount. Note that retryMaxCount is optional, and if this were omitted, then it would be indefinite.

Inbound onMessage() Retry

In 10.1.3, ESBListenerImpl.onMessage() does not throw any exceptions back to the resource adapter. However, the following two JCA ResourceException flavors could be considered:

For non-recoverable (non-retry) cases:

```
oracle.tip.adapter.api.exception.PCResourceException (extends ResourceException)
```

For recoverable (retry) cases:

```
oracle.tip.adapter.api.exception.PCRetriableResourceException (extends
PCResourceException)
```

Inbound Rejection Handling

As the activation agent (bpel.xml) properties are currently not supported, adapter framework-supported rejection handling is not fully available. However, the default rejection handler is available, which kicks in if no explicit exception handler has been declared.

Fatal Error Handling

Currently ESBListenerImpl.java simply incurs JCA EndpointDeactivation() when ESBListenerImpl.onFatalError() is invoked, that is, the resource adapter work thread is terminated (event polling stops).

Alerts

Currently, ESBListenerImpl.java does not do anything when ESBListenerImpl.onAlert() is invoked. The onAlert() API is intended to allow an inbound resource adapter to communicate important and/or critical state changes to an operator console, thereby enabling timely manual operator intervention, to, for example, restart a database or some other EIS facility.

Features Not Supported in Oracle Enterprise Service Bus

The following is a list of features that will not be supported in 10.1.3, Oracle Enterprise Service Bus:

- Implicit correlation
- Streaming of large payloads
- Inbound and outbound request/reply
- Dynamic JCA partner links
- Transparent failover between clustered instances of ESBListenerImpl
- Inbound XPath filtering

5.4.2.3 Connectivity to Oracle Enterprise Service Bus Through Adapter Services

Oracle Application Server adapters provide bidirectional, real-time data access to almost any data source in your enterprise.

An adapter either listens for, or polls for, events in the source application it supports. When listening for events, an adapter registers as a listener for the application that is configured to push events to the adapter. The adapter can also poll the back-end application, such as a database or file, for the events required by Oracle Enterprise Service Bus.

By registering adapters with Oracle Enterprise Service Bus (using a wizard), you integrate external data sources with Oracle Enterprise Service Bus, and finally, with each other.

Currently, Oracle Enterprise Service Bus Server supports the Oracle adapters described in the following table and enables you to define inbound and outbound adapter services for each. An inbound adapter service receives data from an external data source and transforms it into an XML message. An outbound adapter service sends data to a target application by transforming an XML message into the native format of the given adapter.

Adapter Service	Description
AQ adapter service	Sends or receives messages from Oracle Advanced Queuing single or multiconsumer queues
File adapter service	Sends or receives messages from a file in the local file system
FTP adapter service	Sends or receives messages from a file at a remote File Transfer Protocol (FTP) server
Database adapter service	Sends or receives messages extracted from an Oracle Database table or created by executing a stored procedure
JMS adapter service	Sends or receives messages from a JMS queue or topic
MQ adapter service	Sends or receives messages from IBM's MQ queue or topic
Oracle Application adapter service	Sends or receives messages from an Oracle E-Business Suite interface

Any service, except an inbound adapter service, that you create as an Oracle Enterprise Service Bus service, such as an outbound adapter service or routing service, is automatically created as a SOAP service without requiring you to provide configuration details. An inbound adapter service requires you to manually configure it as a SOAP service. Oracle Enterprise Service Bus provides a wizard to take you through a step by step configuration process.

6

Adapter Life-Cycle Management

Oracle Application Server adapters are based on J2EE Connector Architecture (J2CA) 1.5 standards and deployed in the Oracle Containers for J2EE (OC4J). The life cycle of Oracle Application Server adapters depends on Business Process Execution Language for Web Services (BPEL) Process Manager. These adapters integrate with BPEL Process Manager through the Adapter Framework component.

This chapter contains the following sections:

6.1 Installing Oracle Application Server Adapters

Technology adapters and OracleAS Adapter for Oracle Applications are available as part of the Oracle BPEL Process Manager install. In addition, these adapters support both standalone OC4J and middle tier deployments. Packaged-application adapters and Legacy adapters are available as part of the OracleAS Adapters CD. These adapters support middle tier deployment only.

6.2 Starting and Stopping an Adapter

Oracle Application Server adapters are deployed as J2CA 1.5 resource adapters. Therefore, to start or stop an adapter, every resource adapter must implement the `start (BootstrapContext)` and `stop` methods as part of the SPI interface. Oracle Application Server adapters are started when a BPEL process using them starts a J2CA Outbound Interaction. Adapters can also be started when the BPEL process is itself loaded for inbound interactions or when adapters publish events to the BPEL process.

After an adapter is started, the only way to stop the adapter is to shut down OC4J. In this release, Adapter Framework acts as a part of the J2CA 1.5 container.

6.3 Physically Deploying an Adapter

Oracle Application Server adapters are deployed as J2CA 1.5 resource adapters in an OC4J container. Adapters are packaged as Resource Adapter Archive (`RAR`) files using the Java Archive (JAR) format. The physical deployment of adapters involves using the `RAR` file and registering adapters as connectors with the underlying OC4J or the middle tier platform. The `RAR` file contains the `ra.xml` file, which is the deployment descriptor XML file containing deployment-specific information about the resource adapter. In addition, the `RAR` file contains declarative information about the contract between OC4J and resource adapter.

In addition to the `ra.xml` file in the `.rar` file, adapters package the `oc4j-ra.xml` template file. The `oc4j-ra.xml` file is used to define resource adapter `ConnectorFactory` objects (logical deployment). The `oc4j-ra.xml` file is the OC4J-specific deployment descriptor for a resource adapter. It contains deployment configurations for deploying resource adapters to OC4J, which includes the back-end application connection information as specified in the deployment descriptor of the resource adapter, Java Naming and Directory Interface (JNDI) name to be used, connection pooling parameters, resource principal mapping mechanism, and configurations.

Example 6–1 Standalone OC4J install

This example illustrates the physical deployment of technology and OracleAS Adapter for Oracle Applications as a part of the BPEL Process Manager standalone OC4J installation. For the standalone OC4J installation, use the following command to deploy the JCA 1.5 resource adapter:

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

For the standalone OC4J installation, use the following command to remove the JCA 1.5 resource adapter:

```
java -jar $ORACLE_
HOME/integration/orabpel/system/appserver/oc4j/j2ee/home/admin.jar
```

```
ormi://localhost oc4jadmin welcome1 -undeployConnector -name myAdapter
```

The Connectors folder also contains `MANIFEST.MF`, `oc4j-ra.xml`, and `ra.xml` files.

The following is a sample `oc4j-ra.xml` file:

```
<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=""/>
```

Note: For the middle tier installation, run the `dcmctl` utility to deploy the J2CA 1.5 resource adapter. Use the following command:

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

6.4 Logically Deploying an Adapter

The logical deployment of adapters refers to the creation of `ConnectionFactory` objects in `oc4j-ra.xml` deployment descriptor file. To add the connection information and assign to a JNDI name, edit the corresponding `oc4j-ra.xml` file of the resource adapter. In addition, you must manually locate and edit `oc4j-ra.xml` for the required adapter. The `oc4j-ra.xml` file contains run-time connection parameters for an adapter.

For example, to deploy OracleAS Adapter for Database logically, edit the `oc4j-ra.xml` file at the following location:

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

6.5 Viewing Adapter Logs

You can view the logs for Oracle Application Server adapters as follows:

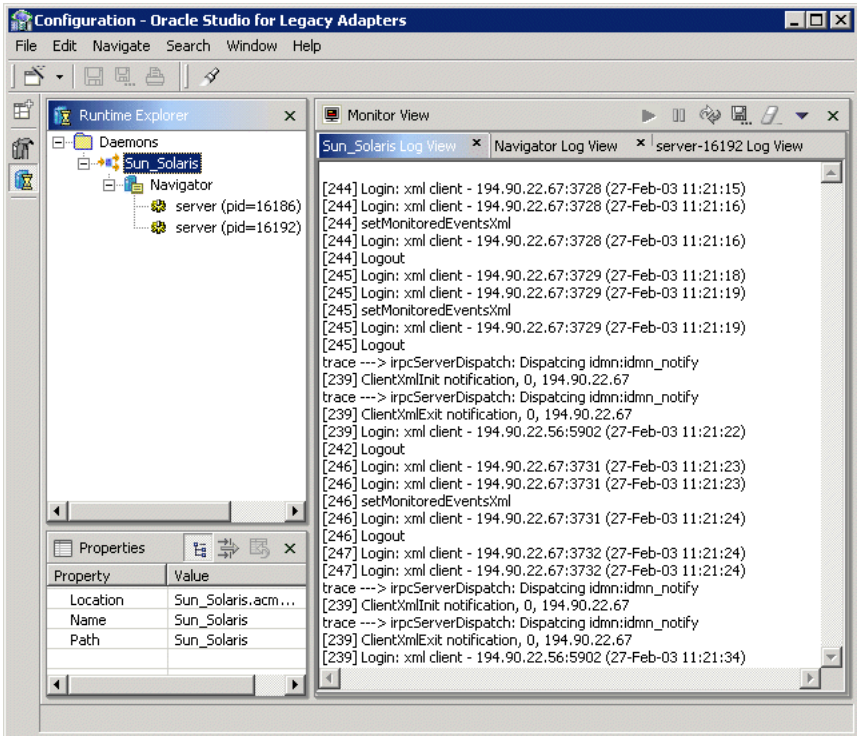
- Technology adapters and OracleAS Adapter for Oracle Applications: These adapters implement the `LogManager` interface of Adapter Framework, which redirects the adapter logs to the BPEL domain logs. For both outbound and inbound interactions, the log files are redirected to BPEL domain logs.

  The BPEL domain logs can be present at the following location:

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

  Note: You can also view adapter logs from BPEL Console.

- Packaged-application: These adapters do not implement the `LogManager` interface because it is not part of the J2CA 1.5 standard. Therefore, the log outputs of these adapters are redirected to the `opmn/logs` directory. For Outbound Interactions, the logs are directed to `opmn` logs. On the other hand, for Inbound Interactions, logs are redirected to the BPEL domain logs.
- Legacy adapters: In addition to the J2CA resource adapter, legacy adapters consists of Oracle Connect, which consists of native adapters for communicating with the mainframe application and data stores. Oracle Connect logs can be viewed using Oracle Studio, which is the mainframe adapter design-time tool and Oracle Connect management tool. Oracle Connect generates various types of logs, such as the daemon log, workspace log, and server process log. The following figure shows the daemon log monitor, which displays activity between clients and the daemon, including clients logging in and out from the daemon.



6.6 Using Adapter Headers

Adapters expose the underlying back-end operation specific properties as Header elements and allow the manipulation of these elements within a business process. For

example, the Header element for OracleAS Adapter for Advanced Queuing exposes Header properties, such as Correlation ID and Message ID, for correlating two Advanced Queuing messages within a business process.

6.7 Setting the Trace Level of an Adapter

Set the trace level for Oracle Application Server adapters as follows:

- Technology adapters and OracleAS Adapter for Oracle Applications: Using BPEL Console, set the trace level for technology adapters and OracleAS Adapter for Oracle Applications. After opening the BPEL Console, click **Manage BPEL Domain**, and then select the **Logging** tab. For outbound interaction, set the `default.collaxa.cube.ws` property. For inbound interaction, set the `default.collaxa.cube.activation` property.
- Packaged-application adapters: For outbound interactions, set the `Loglevel` property for packaged-application adapters in the `oc4j-ra.xml` file.
- Legacy adapters: The trace level for Oracle Connect, the mainframe server, can be set by using Oracle Studio.

6.8 Adapter Deployment Validation Within JDeveloper

During the deployment step (JDev or obant) the WSDL's containing J2CA bindings will be scrutinized by a J2CA WSDL Validation Service. The WSDL Validation Service will only select the WSDLs containing J2CA bindings.

The following validation steps occur:

- Integrity of WSDL (JCA) Bindings
- Validation of JNDI location if it is deployed through JDev, or managedConnectionFactory/ConnectionSpec (if provided in `jca:service`)
- Existence of InteractionSpec or ActivationSpec specified in `jca:operation`
- Validation of all property names and values bound to the InteractionSpec or ActivationSpec in `jca:operation`
- Validation of any XML Record converter specified in jca:binding or `pc:inbound_binding`
- Finally, if the InteractionSpec or ActivationSpec implements the DeploymentValidation interface, its `validate()` method will be invoked

6.9 Turning on the XML Validation

XML validation can be turned on for the BPEL server domain through BPEL Console or at the `PartnerLink` level. The inbound and outbound XML validation is turned on in the BPEL Process Manager.

6.10 Describing XML Data Structure

The Record implementation for Oracle Application Server adapters is XMLRecord. All adapter interactions are started with an XMLRecord. Each JCA record must be an implementation of `oracle.tip.adapter.api.record.XMLRecord`. Each instance of XMLRecord contains one or two instances of RecordElements: one mandatory Payload RecordElement and one optional Header RecordElement. RecordElements

contain data. In addition, each RecordElement contains one BLOB of data, which can either be an UTF-8-encoded XML string or a binary opaque byte stream.

XMLRecord consists of the following methods:

- `getHeaderRecordElement`: Retrieves the Header RecordElement.
- `getPayloadRecordElement`: Retrieves the Payload RecordElement.
- `setHeaderRecordElement`: Sets the header record element of the XMLRecord.
- `setPayloadRecordElement`: Sets the payload record element of the XMLRecord.

6.11 Encrypting Passwords in oc4j-ra.xml

Use the `encrypt` command in the `orabpel\bin` directory (`encrypt.sh` or `encrypt.bat`) to encrypt passwords in the `oc4j-ra.xml` file.

Each JCA resource adapter implementation is then responsible for using the proper Adapter Framework decryption facility to decrypt the password. If the adapter does not do this, then it does not make sense to encrypt any of its `oc4j-ra.xml` entries.

6.12 Managing Errors

The Adapter and the underlying Adapter Framework (AF) library throw the following type of exceptions:

- WSIFException: The JCA artifacts for both the inbound and outbound interactions are captured in the WSDL that defines the Adapter Service. At run-time, the JCA WSIF Provider component of the AF parses the WSDL for JCA artifacts and creates InteractionSpec for outbound interaction. If the InteractionSpec class cannot be located in the class path, or if the instantiation fails, then the AF throws `WSIFException`.
- ActivationException: The BPEL Process throws `ActivationException` on failure of activation of the Adapter endpoint associated with JCA inbound interaction.
- PCRetriableResourceException: The Adapter throws `PCRetriableResourceException` for transient connection errors, which are recoverable errors in case of outbound interaction. The PCRetriableResourceException exception can be retried by defining a retry policy in `bpel.xml`.
- ResourceException: The `ResourceException` exception is thrown in all cases.

6.13 Handling Connection Errors

Technology and OracleAS Adapter for Oracle Applications adapters can handle connection errors for the following interactions:

- Outbound Interaction: technology adapters and OracleAS Adapter for Oracle Applications raise `oracle.tip.adapter.api.exception.PCRetriableResourceException` for transient connection errors, which are recoverable connection errors. For example, a database listener may not have started and this might be giving connection errors.

You can define the maximum number of reconnection attempts that can be made in the `bpel.xml` file. In this file, under the `PartnerLinkBinding` parameter, specify the parameters for reconnection attempts as shown in the following example:

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

In this example, the reconnection parameter settings specify the BPEL run time to make an attempt to reconnect every 10 seconds and allow the maximum number of 30 attempts to reconnect. After the maximum number of attempts, the `RemoteFault` exception is raised for the BPEL process.

All other exceptions cannot be recovered and result in throwing the `BindingFault` exception to the BPEL process.

Note: Packaged-application adapters, except OracleAS Adapter for Oracle Applications, and Legacy adapters return connection exception errors. For these adapters, you need to provide an Exception Filter (Java class) which can translate third party exceptions to either `PCRetriableResourceException` (`RemoteFault`) or `PCResourceException` (`BindingFault`). The exception filter must be declare in the `jca:binding` element, as follows:

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

The filter (Java class) must implement the interface, `oracle.tip.adapter.api.exception.ExceptionFilter`.

- Inbound interaction: Technology adapters, Oracle AS adapters, and legacy adapters support a poll model for connecting to the back-end application for receiving events. In case of unrecoverable connection failures, the adapters recycle old connections, and send out alerts or notifications to the BPEL process. In addition, the adapter shuts down the inbound endpoint and the associated BPEL process instance. The inbound interaction connection errors are written to a log, and can be viewed through the BPEL console.

6.14 Handling Adapter Data Errors

You can handle errors that might occur during the following interactions:

- Outbound interaction: The J2CA Adapter throws a `ResourceAdapter` exception for the Outbound Interaction case. The correction of data and replaying of the message has to be handled within the BPEL Process.
- Inbound interaction: Each J2CA adapter can have one or more Message Rejection handlers associated, as well as a fail over BPEL process in the case of a termination condition. In `bpel.xml`, you can configure how these messages should be handled as follows:

- Write the error to a specific directory
- Write the error to an Advanced Queue (AQ) queue
- Send the error to an error handler BPEL process
- Call a WSIF service

6.15 Describing Message Ordering

Message ordering can be achieved by making a BPEL process synchronous, as it implies that the resource adapter thread publishing a message to the BPEL engine will be used for executing the entire BPEL instance.

For asynchronous BPEL processes, a pool of BPEL engine worker threads will process received messages. Thus, it will prevent a message ordering guarantee, as the duration of one BPEL instance will almost always be different from that of others.

Generally, a JCA resource adapter wizard only creates a one way (asynchronous) WSDLs, that is, WSDLs where the operation has only an input message. It is necessary to manually modify the wizard generated WSDLs, that is, changing it so that it becomes a request-response (synchronous) type WSDL with input and output messages.

Consider the following example to create a XML schema type for a response (output) message that is generally not generated:

```
<types>
  <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>
```

After creating an XML schema type, the next step is to define the WSDL message (in the adapter WSDL file) as shown in the following example:

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

The next step is to add an output message to the inbound WSDL operation as shown in the following example:

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

Next, you need to add an output element in the binding section as shown in the following example:

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

```
    <jca:operation .../>
    <input/>
    <output/>
  </operation>
</binding>
```

The WSDL is OK.

Next, in the BPEL Process you should add a *reply* activity at the end of the business process logic as shown in the following example:

```
<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>
```

If the resource adapter in the preceding example supports multi-threading inbound, it must be configured/tuned to only use one thread, because multiple threads will break the message ordering guarantee due to their stochastic durations.

6.16 Describing Message Rejection Handlers

You can configure the Message Rejection handler to do the following:

- Process translation errors
- Take corrective action
- Issue an alert or a notification message to the BPEL process

You should specify the `Message Rejection` handler in the `bpel.xml` file under the associated `PartnerLink` activity. In addition, you should define Message Rejection handlers as part of the Activation agent properties. The following are four types of Message Rejection handlers:

- File system based Message Rejection handler: The file system based Message Rejection handler writes the bad messages to the configured directory by using the following file name pattern:

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

  The syntax for specifying a file system based Rejection handler is as follows:

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

An example for specifying file based Rejection handler is as follows:

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

- RAW Oracle Advanced Queue based Message Rejection handler: The RAW Oracle Advanced Queue based Message Rejection handler allows the user to designate a Oracle RDBMS RAW AQ queue as the rejection storage.

  An example for specifying RAW Oracle Advanced Queue based Message Rejection handler is as follows:

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

  Ensure that you type `:` and `|` in the exactly the same location in the code as shown in this example.

- BPEL Process Message Rejection Handler: The BPEL Process Message Rejection handler sends the bad message to the designated error handling BPEL process.

  The syntax for specifying a BPEL Process Message Rejection Handler is as follows:

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

  Note: `[:<password>]` in the preceding syntax can be encrypted by using the `encrypt.bat/encrypt.sh` utility.

  You can thus define a process with a receive operation of your choice by choosing (WSDL and BPEL source wise) - the only constraint is on the *WSDL Message Type* of the message that will be sent to this rejection handler. It must be declared to have the type `RejectedMessage`. You can do this by importing the `xmllib` resident `WSDL RejectionMessage.wsdl`, which defines such a message as shown in the following example:

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

  You can import an `xmllib` WSDL from another WSDL by using the URL shown in the following example:

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

  The BPEL process would only contain the import as shown in this example. The port type would reference the following:

  ```
  <definitions ...
  xmlns:rej="http://xmlns.oracle.com/pcbpel/rejectionHandler"
  ```

```
<portType name="MyRejectionHandlerPortType">
   <operation name="myHandleRejectionOperation">
   <input message="rej:RejectionMessage"/>
   </operation>
</portType>
```

- WSIF Based Rejection Handler: In WSIF based Message Rejection handler, you can configure any type of WSIF WSDL, such as JCA, EJB, JMS, HTTP, and Java. You can configure any kind of service that can be reached through WSIF as the bad message handler.

  The syntax for specifying a WSIF based Message Rejection handler is as follows:

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

  You can specify a WSIF based Message Rejectionhandler as shown in the following example:

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

  The WSIF based Message Rejection handler has the same constraint on message type as that of BPEL Process Rejection Handler. Refer to the BPEL Process Rejection Handler section.

6.17 Describing How Adapters Ensure no Message Loss

This section describes how Adapters ensure no message loss, and how you can recover recover messages stuck in the BPEL dehydration store.

The BPEL engine is constructed to always ensure delivery. Dehydration points are established before each *receive*, *pick*, *wait* and after each *reply* and *invoke*. By default dehydration after an invoke is deferred but you have control of that through tuning process parameters (see the `idempotent` setting option in the BPEL Tuning Guide).

The JCA Resource Adapters extend the reach of the BPEL engine and in a specific way contribute to making sure that the delivery guarantee is upheld.

Transactional adapters allow the EIS to participate in one-phase or two-phase commits (local transactions or global/distributed transactions). These commits are controlled either by the adapter (inbound) or by the BPEL engine (outbound). Non-transactional adapters implement their own schemes to ensure delivery, without the use of transactional semantics.

6.17.1 Local transaction and Global (XA) transactions

In BPEL PM 10.1.3 transactional adapters support global (two-phase commit) transactions, through the JCA 1.5 XA contracts. This include adapters for Oracle Applications, Database, Advanced Queuing, JMS and MQSeries. Non transactional adapters include the File and FTP adapters.

6.17.2 Inbound Transactions

For an asynchronous BPEL process, a transactional adapter will initiate a global JTA transaction before sending an inbound message to BPEL. The inbound message is

queued into the dehydration store through the Delivery Service in the BPEL engine. When control returns to the adapter, it will commit the JTA transaction thus executing the following set of actions as an atomic unit of work:

1. commit the removal of the message from the inbound adapter endpoint (for example, table and queue.)
2. commit the insertion of this message to the dehydration store.

If anything fails during this, both the actiond 1 and 2 will be rolled back guaranteed.

After successful message delivery by the adapter to the dehydration store, the inbound message remains in the store until the next BPEL process dehydration point occurs, for example, until the first invoke. At that point the message from the receive activity is removed from the Delivery Service. Any activities between these checkpoints are treated as being part of a JTA (global) transaction. If the BPEL server fails between the two dehydration points, the entire transaction is rolled back so that state is assured. Your last message would be replayed by the next available BPEL server. All this happens under the covers.

For a synchronous process, the global transaction initiated by the adapter will span message delivery and BPEL process execution up until the first reply activity, which is typically placed at the end of a process flow.

6.17.3 Outbound Transactions

For transactional adapters outbound JCA interactions (the invoke activities) are scoped with the global BPEL JTA (ejb) transaction. This means that all BPEL activities, including JCA adapter invocations will be part of a global transaction, and as such either all activities are committed or rolled back, if an error occurs.

For example, a BPEL process can insert data into several tables (on different databases) through different invoke activities (invoking the Database adapter). When the BPEL instance is about to finish, the JTA transaction is committed. Only at that point will the database insert operations be committed. If any errors occur during the BPEL instance execution, all activities (and thus database operations) will be rolled back to the last dehydration point.

6.18 BPEL Clustering Support Within Adapters

The Adapter Framework supports active fail-over of inbound Adapter Services. You can achieve this by adding a property to a particular JCA activation agent (in bpel.xml) as shown in the following example:

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

If the BPEL PM servers (JVMs) in the cluster are located across TCP/IP subnet boundaries, then it is necessary to add the attribute `clusterAcrossSubnet=true`.

In a cluster group, the multiple activations of the same (for example, File) adapter Activation Agent (for a specific partnerlink) will be detected implicitly and automatically by all the instances of the adapter framework active in that cluster. Only one activation will be allowed to actually start the reading or publishing messages. The adapter framework instances chooses one among them, randomly as to who should assume the Primary Activation responsibility. The other activations (instances) in the cluster will initiate to a hot stand-by state, without actually invoking *EndpointActivation* on the JCA resource adapter.

If a primary activation at some point becomes unresponsive, is deactivated manually or if it crashes/exits, then any one of the remaining adapter framework members of the cluster group will immediately detect this, and reassign the primary activation responsibility to one of activation agents standing by. This feature uses JGroups underneath for the implementation, hence the *clusterGroupId* property.

6.19 Deploying Adapter Services

Adapter services can be deployed in two ways, using JDeveloper and BPEL Console. In addition, deploying a BPEL process deploys an adapter service. You can remove an adapter service from the BPEL Console.

6.20 Batching and Debatching Support

The batching and debatching functionality is supported only by OracleAS Adapter for Files and OracleAS Adapter for FTP, and OracleAS Adapter for Databases. OracleAS Adapter for File and OracleAS Adapter for FTP consist of a Reader to debatch a single huge file into several batches. You need to specify the batch size during the design-time configuration. In addition, the adapter includes a Writer to batch a set of messages into a single file.

OracleAS Adapter for Databases consists of a Publish component to poll a set of tables to detect events. This component can raise events to the BPEL process one record at a time or multiple records at a time.

6.21 Migrating Repositories

All the adapter service WSDLs generated by the Adapter Configuration wizard have a reference to the JNDI name. The reference is defined in the `oc4j-ra.xml` file which is the adapter's deployment descriptor, through the location attribute on the `jca:address` element. This is the *key* when you want to migrate from a development environment to a test environment to a production environment. You should update the `oc4j-ra.xml` file to have the same JNDI name in all three environments: development, testing, and production. You should specify values for deployment time properties, such as retry interval and retry count, and then redeploy to testing environment or production environment. The `oc4j-ra.xml` will identify the end point as a development EIS or testing EIS or production EIS. For example, consider that when running through the Database Adapter Service wizard, you specify `eis/DB/custStore` as the JNDI name for `createCustomer` service. After modeling the BPEL process by using this adapter service, you should deploy it to the development, test or production environments without making any changes. But before you do this, ensure that you have a corresponding JNDI entry for `eis/DB/custStore` in each of your various environments pointing to the right EIS instance.

Index

T

W