BEA WebLogic Integration™

Introducing Application Integration

Version 8.5 Service Pack 5
Revised: October 2005

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

1. Introduction to Application Integration

2. Understanding Application Integration

3. Roles, Responsibilities, and Tasks

4. Understanding the ADK

5. Understanding the Development Kit Adapters

Index

CHAPTER 1

Introduction to Application Integration

WebLogic Integration provides a standards-based integration solution for connecting applications both within and between enterprises. WebLogic Integration provides the following tools for integrating applications:

- Adapters
- Application Views
- Application View Controls
- Adapter Development Kit (ADK)

By using these tools, you can integrate all your enterprise information systems. Typical IT organizations use several highly specialized applications. Without a common integration platform, integration of such applications requires extensive, highly specialized development efforts.

Adapters

WebLogic Integration makes use of adapters to establish a single enterprise-wide framework for integrating current or future applications. Adapters greatly simplify your integration efforts because they allow you to integrate each application with a single application server, and thus avoid the need to integrate every application with every other application.

The BEA WebLogic Adapters for WebLogic Integration enable fast, simple, and robust enterprise application integration. Compliant with the J2EE™ Connector Architecture specification from Sun Microsystems, Inc., each adapter provides bi-directional,

request-response integration with a specific application, protocol, or technology. You purchase the individual adapters you want, and then install them to work in conjunction with BEA WebLogic Integration. User information on specific adapters is available at http://edocs.bea.com. Please check the BEA web site or contact Customer Support for platform support information.

The BEA WebLogic Adapters for WebLogic Integration provide:

- Easy and Fast Integration with Enterprise Information Systems
- Compliance with the J2EE Connector Architecture
- Scalable, Reliable, and Secure Integration Framework

In addition to the BEA WebLogic Adapters for WebLogic Integration, developers can create custom adapters using the WebLogic Integration Adapter Development Kit (ADK). The ADK is a set of tools for implementing the events and services supported by BEA WebLogic Integration. The process of creating custom adapters with the ADK is described in *Developing Adapters* at the following URL:

`http://edocs.bea.com/wli/docs81/devadapt/index.html`

Note: Throughout the rest of this document, the term *adapter* refers to any of the BEA WebLogic Adapters for WebLogic Integration, while the term *custom adapter* refers to any custom adapter that was created using the ADK.

Easy and Fast Integration with Enterprise Information Systems

An *enterprise information system* (EIS) is an application that provides the information infrastructure for an enterprise. An EIS offers its clients a set of services that are made available to clients via local and/or remote interfaces. Examples of EISs include:

- Enterprise Resource Planning (ERP) systems, such as SAP R/3 or PeopleSoft
- Customer Relationship Management (CRM) systems, such as Siebel
- Database systems, such as Oracle

The BEA WebLogic Adapters for WebLogic Integration work with the most popular and widely used EISs. The adapters enable organizations to quickly design and deploy integration solutions that involve existing and future EIS resources. Adapters simplify integration efforts by providing robust, standards based connectivity with various applications within a coherent framework built on top of WebLogic Server.

Compliance with the J2EE Connector Architecture

The BEA WebLogic Adapters for WebLogic Integration are implementations of the *J2EE Connector Architecture* (JCA) version 1.0, from Sun Microsystems, Inc. The JCA is used for integrating J2EE-compliant application servers with enterprise information systems (EIS). The JCA consists of two parts: an EIS-specific resource adapter (such as those provided in the BEA WebLogic Adapters for WebLogic Integration) and an application server (such as BEA WebLogic Server) that the resource adapter plugs into.

The JCA defines a set of contracts, such as transactions, security, and connection management, that a resource adapter must support in order to plug in to an application server. For more information, see the Sun JCA page at the following URL:

`http://java.sun.com/j2ee/connector/`

Scalable, Reliable, and Secure Integration Framework

The BEA WebLogic Adapters for WebLogic Integration are crucial components of a comprehensive enterprise integration framework that provides:

- *Scalability* via the clustering, load balancing, and resource pooling features of BEA WebLogic Platform. In a scalable deployment, adding a linear amount of resources—such as memory, processors, or machines—will result in a corresponding linear increase in throughput while maintaining the same response level.
- *Reliability* via the fault-tolerant and server fail-over features of BEA WebLogic Platform. If a critical problem occurs—such as an e-business application bug or an operating system, hardware, or network failure—integration solutions remain up-and-running.
- *Security* for mission-critical applications and data via the extensive security mechanisms of BEA WebLogic Platform as well as the security of integrated EISs. WebLogic security mechanisms provide user authentication and authorization capabilities, integration with existing security systems (such as LDAP) through security realms, controls in both the presentation and business layers, and firewall interoperability.

Application Views

WebLogic Integration helps you use adapters to define business-focused interfaces to an EIS. These interfaces, called *application views*, provide a simple, consistent, self-describing interface to services and events in an application. Unlike adapter users, application view users are not required to have intimate knowledge of the EIS or its client interface. As a result, users who are not programmers, such as technical analysts, can use application views.

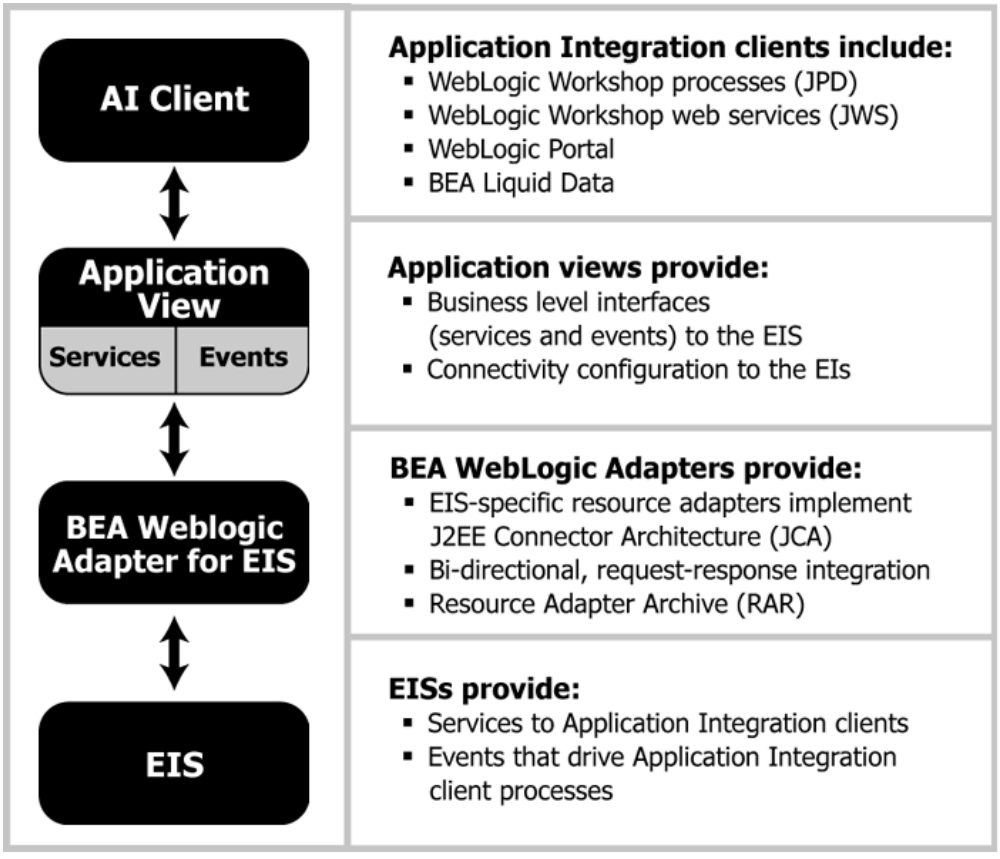
Application View Control

You use application view controls in WebLogic Workshop to interact with an EIS through an application view. Application view controls allow a business process engineer to browse the hierarchy of application views, invoke a service as a business process action, and start a new business process when an EIS event occurs.

How Do Adapters Fit Into the WebLogic Architecture?

Adapters are used in conjunction with the application integration capabilities of BEA WebLogic Integration. These capabilities provides a systematic, standards-based architecture for hosting business-oriented interfaces to enterprise applications.

The following illustration shows how the various application integration components interact.


AI Client
Application View
Services
Events
BEA Weblogic Adapter for EIS
EIS
Application Integration clients include:
WebLogic Workshop processes (JPD)
WebLogic Workshop web services (JWS)
WebLogic Portal
BEA Liquid Data
Application views provide:
Business level interfaces (services and events) to the EIS
Connectivity configuration to the EIs
BEA WebLogic Adapters provide:
EIS-specific resource adapters implement J2EE Connector Architecture (JCA)
Bi-directional, request-response integration
Resource Adapter Archive (RAR)
EISs provide:
Services to Application Integration clients
Events that drive Application Integration client processes

CHAPTER 2

Understanding Application Integration

The application integration capabilities provided by WebLogic Integration offer a standards-based architecture for hosting *application views*: business-oriented interfaces to enterprise applications.

This section includes the following topics:

The Application Integration Lifecycle

The various application integration components participate in the following high-level lifecycle:

1. Define the overall integration solution. This includes defining what EIS and adapters are used and what services and events are implemented.
2. Install and deploy the required adapters.
3. Create a WebLogic Workshop application that implements the required business processes for the integration solution.

4. Define an application view that addresses a specific business purpose. This step includes defining the required services and events and testing the application view.
5. Publish the application view to the WebLogic Workshop application.
6. Define an Application View control that provides access to application view services.
7. Integrate the Application View control into your business process.
8. Deploy your integration solution.
9. Manage your integration solution using the WebLogic Integration Administration Console.

Understanding Adapters

A *resource adapter* (or simply *adapter*) is a software component that acts as a connector between an EIS and a J2EE application server (such as BEA WebLogic Server). Each adapter provides bi-directional, request-response integration with a specific application or technology. Resource adapters are implementations of the *J2EE Connector Architecture* (JCA) version 1.0, from Sun Microsystems, Inc. For more information, see the Sun JCA page at the following URL:

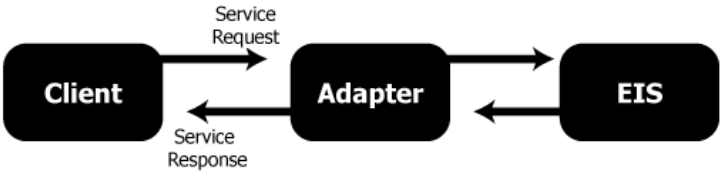
`http://java.sun.com/j2ee/connector/`

Application integration uses adapters and associated application views to help you integrate applications in your enterprise. Instead of *hardwiring* your enterprise systems together, you can use adapters to connect enterprise systems to an application server. Once you deploy an adapter for an EIS, other components and applications can use that adapter to access data on the EIS.

Supported Operations

Adapters handle two general types of operations:

- *Services* are request / response communications with the EIS. Client applications submit service requests to the EIS via the adapter, and the adapter returns the EIS response back to the client. For example, a business process might invoke a SAP BAPI or execute a SELECT statement on a database. Responses are either synchronous or asynchronous.



- *Events* are asynchronous, one-way messages received from an EIS. For example, the adapter can receive an IDOC from a SAP system or a message from an MQ system. The adapter routes the EIS message to the appropriate software component via the WebLogic Integration Message Broker and the Application Integration JMS infrastructure.

In effect, a service is a *request for some work to be done* and an event is a *notification that some work has been done*.

At run time, the EIS and the adapter exchange requests, responses, and events as XML documents. The adapter automatically handles the data translation between the EIS format and the XML format via schemas that are defined at design-time.

An adapter instance defines zero or one event connection and zero or more service connections. Each adapter instance is related to a base adapter which is deployed as a RAR file.

Understanding Application Views

An *application view* is a business oriented interface to objects and operations within an EIS. Application views include the information needed to communicate with the EIS as well as configurations for services and events. Application views define:

- **Communication with the EIS**, including connection settings, login credentials, and so on.
- **Service invocations**, including the information that the EIS requires for the request, as well as the service request and response schemas associated with the service.
- **Event notifications**, including the information that the EIS publishes and the event schemas for inbound messages.

An application view is typically configured for a single business purpose and contains only the services or events required for that business purpose. An EIS might have multiple application views defined for different business purposes. For example, an EIS containing human resources data might have an *HREmployee* application view that provides individual employees with read-only access to their personnel information, and an *HRDataEntry* application view that provides data entry clerks with the ability to add, update, and delete personnel information.

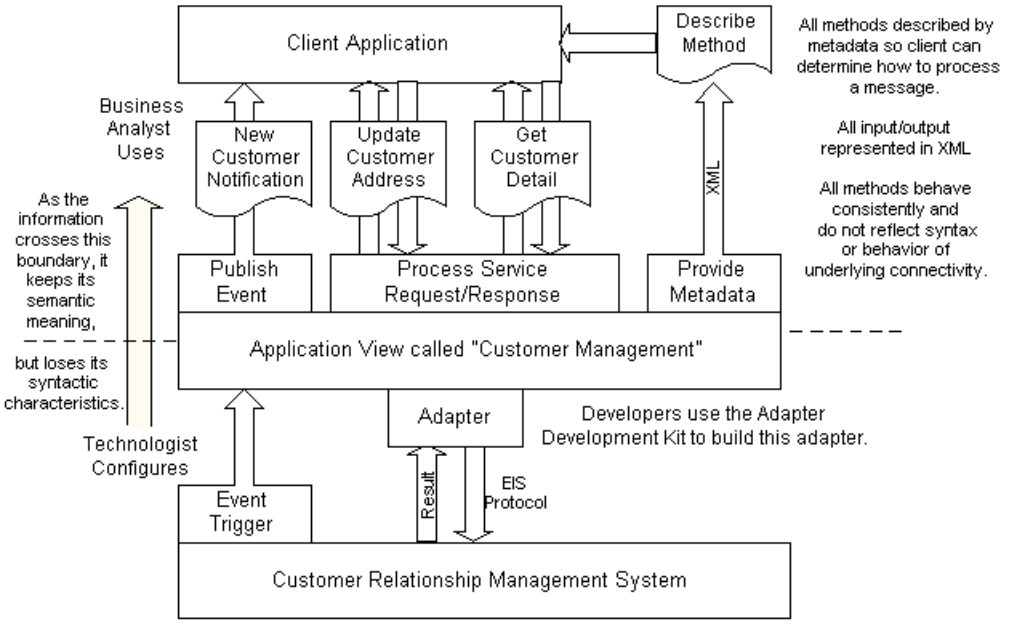
Application views provide a layer of abstraction between an adapter and the EIS functions exposed by that adapter. By using application views, you can simplify the procedure you must perform to access adapters. Instead of accessing an EIS by directly invoking it, you can simply edit the adapter's application views, create new application views, or delete obsolete ones. This layer of abstraction, formed by application views, makes it easy for nonprogrammers to maintain the services and events exposed by the adapter.

Each application view defines a set of business functions on one adapter's EIS. After an adapter is created, you can use its Web-based interface to define your own application views. Such application views allow you to display the application capabilities exposed by an adapter.

If you are a business analyst or technical analyst and you define an application view using an adapter, you can customize the application view for a specific business purpose. The business purpose is defined by the business analyst. For example, if you define a "Customer Management" application view on an adapter for a CRM (Customer Relationship Management) system, then you are likely to add only services and events that are related to customer management. You can, however, create application views that are as inclusive as necessary. Because application views can be customized for a specific business purpose, they work much better than the "one size fits all" approach used by many other EAI systems.

The business-level view of an application's capabilities provides a logical dividing line between the focus of the programmer and that of the technical analyst. For example, with a business-level view, a technical analyst can create records in a database without knowing SQL. Figure 2-1 provides a diagram of an application view at work in an application integration environment.

Figure 2-1 How Application Views Work



You create application views using the Application Integration Design Console, which is described in "Application Integration Design Console" on page 2-14. For detailed information about application views, see Defining an Application View in *Using the Application Integration Design Console* at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/2usrdef.html`

You can use application views from a business process (JPD), web service (JWS), or from BEA Liquid Data. You can also write custom code to access an application view. For more information, see Using Application Views by Writing Custom Code in *Using the Application Integration Design Console* at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/4usrcust.html`

Main Features of Application Views

Because WebLogic Integration uses the application view as its primary user interface for adapters, it offers several features not commonly found in competing EAI technologies:

- Use of XML as a Common Language Among Applications
- Use of Service and Event Definitions to Expose Application Capabilities
- Use of XML Schemas to Define the Data for Services and Events
- Support of Bidirectional Communication in Adapters

The remainder of this section provides descriptions of these functions.

Use of XML as a Common Language Among Applications

In an EAI scenario, it is much easier and more efficient to use one common data format to integrate every EIS with WebLogic Server than it is to use a variety of custom, proprietary data formats to integrate each EIS with every other EIS. When a common data format is used, all applications communicate using a standard language. WebLogic Integration uses XML, the increasingly popular data interchange format, as its common data format.

In the WebLogic Integration environment, virtually all messages are sent as XML documents:

- For each service, application views require an XML request message and provide an XML response message.
- When events are generated, registered event listeners receive event information as XML messages. The application view relies on its adapter to translate the EIS-specific format into and from XML.

Because an adapter translates an application's data format using XML, business analysts do not need to understand that format themselves. If you are a business analyst and you want to use an adapter, you need to know only how to define and use application views. Best of all, because all adapters use a similar Web-based interface for defining application views, it is easy to learn to use current and future adapters. Thus XML simplifies the use of EAI for developers and business analysts alike.

Use of Service and Event Definitions to Expose Application Capabilities

The application view, via an underlying adapter, supports events and services for a particular business use. *Events* enable messages generated by an application to be managed following a publish and subscribe model. *Services* are business functions that may be invoked by a user.

Service invocations cause messages to be sent to an application following a request/response model. Events, service requests, and service responses are all passed through the system as XML documents.

Use of XML Schemas to Define the Data for Services and Events

Each application view uses an XML schema as *metadata*: that is, as information about the XML information for events, service requests, and service responses. This metadata helps users understand the data requirements of any application view event or service.

Support of Bidirectional Communication in Adapters

Currently, the J2EE Connector Architecture Specification version 1.0 does not provide guidelines governing how an EIS initiates communication with an application server or client. WebLogic Integration provides this communication capability via events.

Application Integration Service Clients and Event Consumers

This section describes clients for service invocations and consumers for event notifications.

Clients for Service Invocations

The following table describes the kinds of clients that invoke services on an EIS via an application view.

Table 2-1 Common Service Clients

Client	Description

BEA WebLogic Workshop:
• business processes
• web services
• portals
	Business processes, web services, and portals all access EIS data via the Application View Control, a Workshop control that provides access to an application view and, therefore, to the services defined for the associated EIS. The Application View control allows a business process engineer to browse the hierarchy of application views and to invoke a service as a business process action.
	Synchronous services are represented as simple methods with a single parameter and a non-void return value. For an illustration of synchronous services, see "Processing Synchronous Service Invocations" on page 2-16. Asynchronous services are represented as both a method with a single parameter (the request), and a callback method with a single parameter (the response). For an illustration of asynchronous services, see "Processing Asynchronous Service Invocations" on page 2-18.
	For more information, see the following topics in the BEA WebLogic Workshop Help System:
	• Building Integration Applications
	• Building Web Services
	• Building Portal Applications
	at the following URL:
	`http://edocs.bea.com/workshop/docs81/doc/en/core/index.html`
	In addition, see "Using Applications With Business Processes" in *Using the Application Integration Design Console* at the following URL:
	`http://edocs.bea.com/wli/docs81/aiuser/3usruse.html`

Custom Java Applications	Any Java application that uses the Application View client API (in `com.bea.wlai.client`) can invoke services on an application view. For more information, see "Using Application Views by Writing Custom Code" in *Using the Application Integration Design Console* at the following URL: `http://e-docs.bea.com/wli/docs81/aiuser/4usrcust.html`

Event Consumers

Adapters deliver events using the WebLogic Integration Message Broker, which provides business processes with a channels-based publish and subscribe communication mechanism. For an illustration, see "Processing Event Notifications at Run Time" on page 2-20. The following table describes common consumers of events from an EIS.

Table 2-2 Common Event Consumers

Client	Description

BEA WebLogic Workshop:
• business processes
• web services
• portals
	Business processes, web services., and portals can subscribe to events published by the Message Broker via the Message Broker Subscription control (or, for business processes only, a static subscription). The Message Broker control listens for application view events. Events can start business processes in which the start node is configured with "Started with a Message Broker Subscription".
	For more information, see the following topics in the BEA WebLogic Workshop Help System:
	• Message Broker Subscription Controls in "Using Integration Controls"
	• Building Integration Applications
	at the following URL:
	`http://edocs.bea.com/workshop/docs81/doc/en/core/index.html`
	In addition, see "Receiving Events" in "Using Applications With Business Processes" in *Using the Application Integration Design Console* at the following URL:
	`http://edocs.bea.com/wli/docs81/aiuser/3usruse.html`

Custom Java Applications	Any Java application that uses the Application View client API (in `com.bea.wlai.client`) can consume events. For more information, see "Using Application Views by Writing Custom Code" in *Using the Application Integration Design Console* at the following URL: `http://e-docs.bea.com/wli/docs81/aiuser/4usrcust.html`

Understanding Design-Time GUIs

The design-time capabilities provided by WebLogic Integration provide a means for developers to create the Common Client Interface (CCI) for each adapter. The CCI enables applications components and Enterprise Application integration (EAI) frameworks to drive interactions across heterogeneous EISs using a common client API. An adapter's design-time GUI enables nonprogrammers to rapidly create, deploy, test, and edit application views, which they can customize by adding services and events.

Creating Application Views Using a Design GUI

The primary purpose of an adapter's design GUI is to allow you to define, deploy, and test application views. For detailed information about defining application views, see *Using the Application Integration Design Console*.

Managing Application Views with the Console

The Application Integration Design Console helps you access, organize, and edit all application views in your enterprise. You can use the Application Integration Design Console to create new folders and then add new application views to them. These folders allow you to organize your application views according to your own navigation scheme, regardless of the adapter used by the application view.

For detailed information about managing application views, see "Using the Application Integration Design Console" in *Using the Application Integration Design Console.*

When to Define Application Views and When to Write Custom Code

Using an adapter's design-time GUI is not the only way to expose the functionality of an EIS, but it is usually the most convenient method. To support service invocations and events, you can define application views, or you can write custom code that accomplishes equivalent functions. At a minimum, you must define application views for each adapter, to expose the functions provided by the adapter's application. However, if your users require a greater than average degree of control, you may also write custom code that allows them to access the resources of an adapter. You must decide whether the needs of your enterprise can best be met by defining application views, writing your own code, or implementing a combination of both methods.

When to Define Application Views

Most EIS applications can be integrated into your system easily by defining application views. You may want to define application views in the following situations:

- You have multiple EIS systems in your enterprise, and you lack developers with a detailed, thorough knowledge of them.
- You want to use WebLogic Workshop to construct and manage business processes.
- You need to update the parameters of an adapter or one of its processes.

When to Write Custom Code Instead of Defining Application Views

In general, you should write a custom interface to an adapter only in the following situations:

- You have only one EIS system in your enterprise.
- Your developer has thorough, detailed knowledge of each EIS involved in the business processes being coded.
- You do not need to use the business process capabilities in WebLogic Integration.
- You do not anticipate any need to change your code.

These use cases assume you are using JCA 1.0 adapters directly with WebLogic Server. In these cases, you are coding directly to the JCA CCI. This bypasses the capabilities of application views and WebLogic Integration.

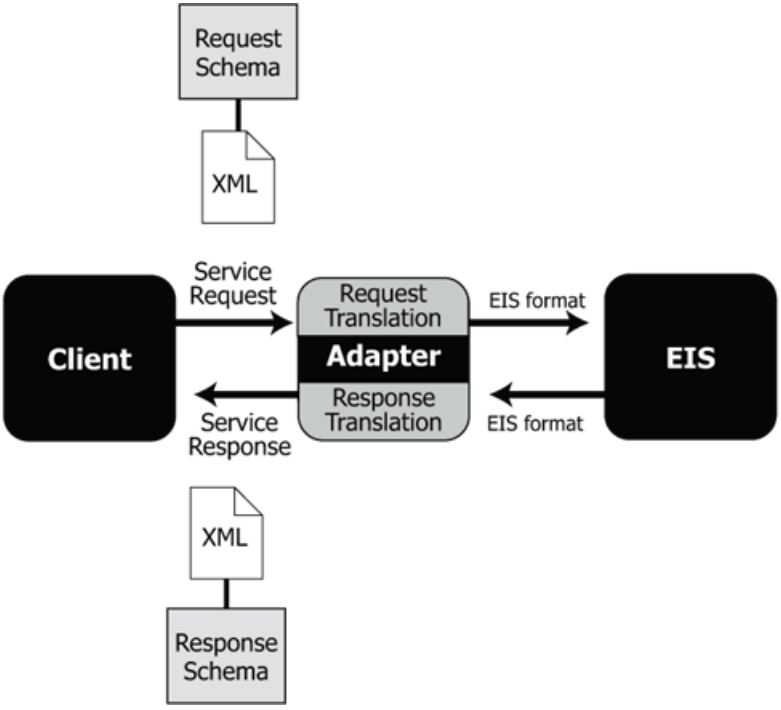
EIS Metadata, Schemas, and Repositories

Each EIS uses its own interface to handle service requests and event notifications. For example, SAP provides a BAPI interface that defines the parameters and syntax for BAPI requests and responses. For each EIS, the EIS interface defines the *metadata* that applications can use to integrate with the EIS. The EIS publishes data and expects requests in the format dictated by its interface rules and metadata.

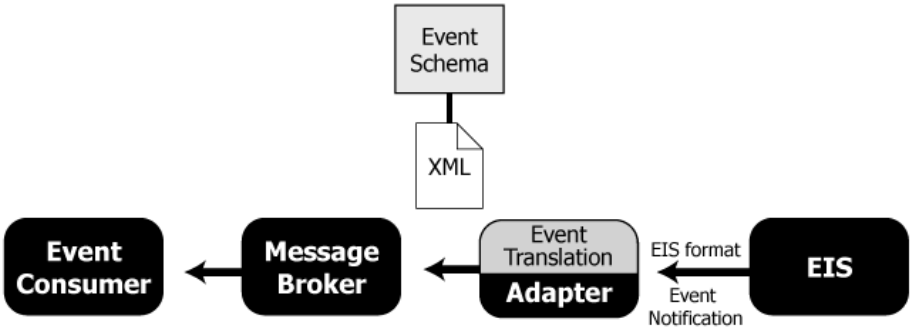
Schemas

At run time, the EIS and the adapter exchange service requests, service responses, and events via XML documents. The adapter handles the data translation between XML documents and the EIS format, using *schemas* that have been defined at design-time to map the data between XML and the EIS format:

- For service requests, the request arrives at the adapter in the form of an XML document. The adapter uses the *request schema* associated with the service (as defined in the application view) to translate the request to the format that the EIS expects. Similarly, when the adapter receives the response back from the EIS, it uses the *response schema* associated with the service to translate the response to an XML document that the requesting application handles.



- For event notifications, the inbound message arrives at the adapter in the format that the EIS uses to publish the event. The adapter uses the *event schema* associated with the event (as defined in the application view) to translate the response to an XML document that the subscribed application handles.



At design time, you define a request and a response schema for each service and an event schema for each event that you configure in the application view. For some adapters, such as SAP, you can use the BEA Application Explorer, which is described in "BEA Application Explorer" on page 2-13. For other adapters, you need to create schemas manually. For instructions on how to define schemas for a particular adapter, see the "Generating Schemas" chapter in the *User's Guide* for the adapter(s) that you are using.

Repositories

Once you have created the necessary schemas, you save them in a file-based *repository*, along with a manifest file that associates the schemas with events and services. When you configure application views in the Application Integration Design Console, you specify the location of the repository so that the application view can find the schemas as needed. For more information, see the "Defining Application Views" chapter in the *User's Guide* for the adapter(s) that you are using.

Tools for Integration Solutions

This section describes the following tools for designing and deploying integration solutions that involve EIS integration:

- BEA Application Explorer
- Application Integration Design Console
- BEA WebLogic Workshop
- WebLogic Integration Administration Console

BEA Application Explorer

Note: The BEA Application Explorer is only used for some BEA WebLogic adapters. Refer to your adapter documentation to see if you need to use this tool.

The BEA Application Explorer is a design-time tool that you can use to generate schemas for services and events. The BEA Application Explorer incorporates in-depth knowledge of application system environments to query for metadata on specific business objects in the EIS. It uses that metadata to generate the schemas required to build the selected service or event—request and response schemas for services and the event schemas for events. For an introduction to schemas, see "EIS Metadata, Schemas, and Repositories" on page 2-11.

For instructions on how to define schemas for a particular adapter, see the "Generating Schemas" chapter in the *User's Guide* for the adapter(s) that you are using.

Application Integration Design Console

The Application Integration Design Console is a design-time tool that you use to build application views and configure services and events. For each event or service, the Application Integration Design Console allows you to configure connection settings and other relevant information.

For instructions on how to create application views using the Application Integration Design Console, see the "Defining Application Views" chapter in the *User's Guide* for the adapter(s) that you are using, as well as "Defining an Application View" in "Introduction to Application Integration" in *Using the Application Integration Design Console* at the following URL:

http://edocs.bea.com/wli/docs81/aiuser/1usrntr.html

BEA WebLogic Workshop

BEA WebLogic Workshop is an integrated development environment for building enterprise-class applications on the BEA WebLogic Platform. WebLogic Workshop is both a design-time tool for building business processes, web services, and portals, and a run-time environment for running business processes.

BEA WebLogic Workshop provides the following mechanisms for integrating with EISs:

- The Application View control lets a developer invoke Application View services both synchronously and asynchronously.
- The Message Broker Subscription control lets event consumers subscribe to Message Broker channels and listen for events that the adapter has received from the EIS and has published via the Message Broker.

  Note: To start business processes based on events, the start nodes should be configured to start with a Message Broker Subscription.

For detailed information, see Starting Your Business Processes in the BEA WebLogic Workshop Help System at the following URL:

http://edocs.bea.com/workshop/docs81/doc/en/integration/wfguide/wfguideStart.html

WebLogic Integration Administration Console

The WebLogic Integration Administration Console allows you to manage deployed application views and adapter instances. For each application view, an administrator can perform management tasks, including the following:

- display and reset event and service statistics
- set environment variables and the security policy
- suspend and resume an application view
- change container managed sign-on settings
- change auto suspend settings
- switch event and service connections

For each adapter instance, an administrator can perform management tasks, including the following:

- display event and service statistics
- display application views using the adapter instance
- suspend and resume an adapter instance
- edit/modify event and service connections
- redeploy adapter instances

For more information on the console, see *Managing WebLogic Integration Solutions*. The information is also provided in the WebLogic Integration Administration Console help.

Run-Time Processing of Services and Events

This section provides a high-level overview of how adapters process services and events at run time. It contains the following topics:

- Processing Service Invocations at Run Time
- Processing Event Notifications at Run Time

The procedures in this section provide simplified, high-level (non-programmer) descriptions of the process. For sample code, see "Code for Sample Java Class" in "Using Application Views by

Writing Custom Code" in *Using the Application Integration Design Console* at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/4usrcust.html`

In these procedures, we use the term *adapter instance*. An adapter instance defines zero or one event connection and zero or more service connections. Each adapter instance is related to a base adapter which is deployed as a RAR file.

Processing Service Invocations at Run Time

Service invocations can be either synchronous or asynchronous:

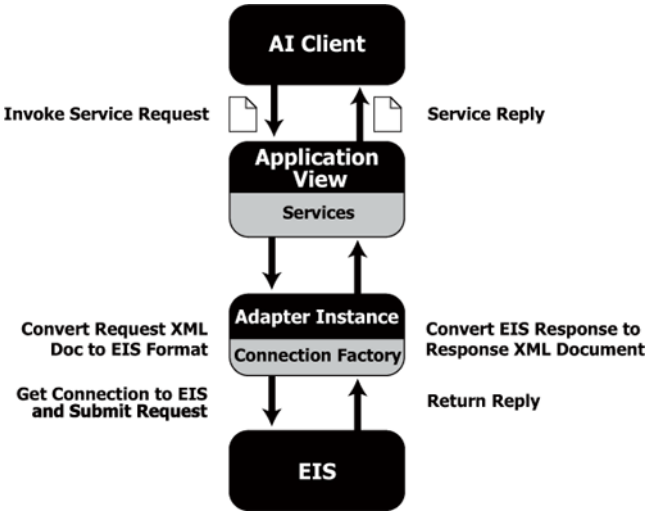
- For *synchronous service invocations*, the client application stops until it receives the response from the EIS.
- For *asynchronous service invocations*, the client application continues, but polls periodically for the response from the EIS or receives the response in a callback method.

The steps for processing service invocations differ when invoked synchronously or asynchronously.

Processing Synchronous Service Invocations

This section walks through the process of a synchronous service invocation at run time, as shown in the following figure.

Figure 2-2 Run-Time Processing of a Synchronous Service Invocation



The following procedure describes, at a high level, how a synchronous service invocation is processed at run time:

1. The client application invokes a given service on a given application view (`invokeService` method), specifying the service name, application name, and the request document.

   The client application specifies the response document as the return value to the `invokeService` method.

2. Based on the service invoked, the client instance of the application view obtains a connection to the EIS from the connection factory defined in the adapter instance, and then establishes a connection to the EIS.

3. The client instance of the application view requests that the adapter execute the service request (`execute` method).

4. Upon receiving the request document, the adapter:

   – Translates the request document into the appropriate EIS format using the request schema that was configured for the service.

   – Submits the request to the EIS for processing using the appropriate communications technology for the EIS.

5. The EIS processes the request and returns the response.

6. Upon receiving the response from the EIS, the adapter:
   - Translates the response to the XML format using the response schema that was configured for the service.
   - Returns the response document to the client.
7. The client receives the response document as the return value to the `invokeService` method and processes it accordingly.

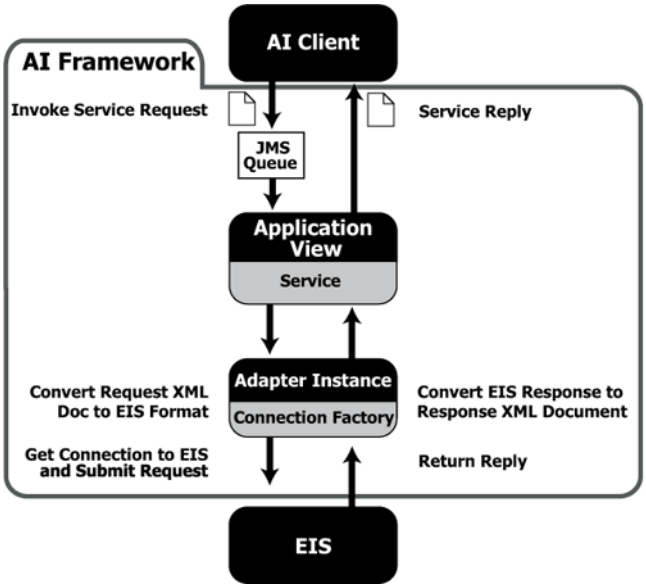
Processing Asynchronous Service Invocations

This section walks through the process of an asynchronous service invocation at run time. Asynchronous service invocations can be initiated in either of two ways:

- If the service is invoked on a business process using an Application View control, then the service has already been configured for asynchronous invocation. At design time in WebLogic Workshop, while configuring an application view control, you specify an asynchronous invocation by selecting the service in the Services to Invoke Asynchronously list in the Application View Browser window.
- If the service is invoked programmatically, then the client application specifies the request ID as the return value to the method indication. The client application also specifies a callback handler method to match a response with the request ID and to receive the response as a response document.

The following figure shows how an asynchronous service invocation is processed at run time.

Figure 2-3 Run-Time Processing of an Asynchronous Service Invocation



The following procedure describes, at a high level, how an asynchronous service invocation is processed at run time:

1. The client application invokes a given service on a given application view, specifying the service name, application name, and the request document.
2. The request document is put into a JMS queue.
3. The application integration framework has a message-driven bean that pulls the request document off the JMS queue and invokes the service on the application view, which, in turn, invokes the adapter.
4. Upon receiving the request document, the adapter completes the following operations:
   - Converts the request document to the appropriate EIS format.
   - Submits the request to the EIS for processing.
5. The EIS processes the request and returns the response.
6. Upon receiving the response from the EIS, the adapter completes the following operations:
   - Retrieves the request ID and matches responses to the appropriate request ID.

  - Translates the response to the XML format using the response schema that was configured for the service.

7. The response document returns to the client application when the adapter returns the response document to the client application via the callback method.
8. The client application receives the response document and processes it accordingly.

Processing Event Notifications at Run Time

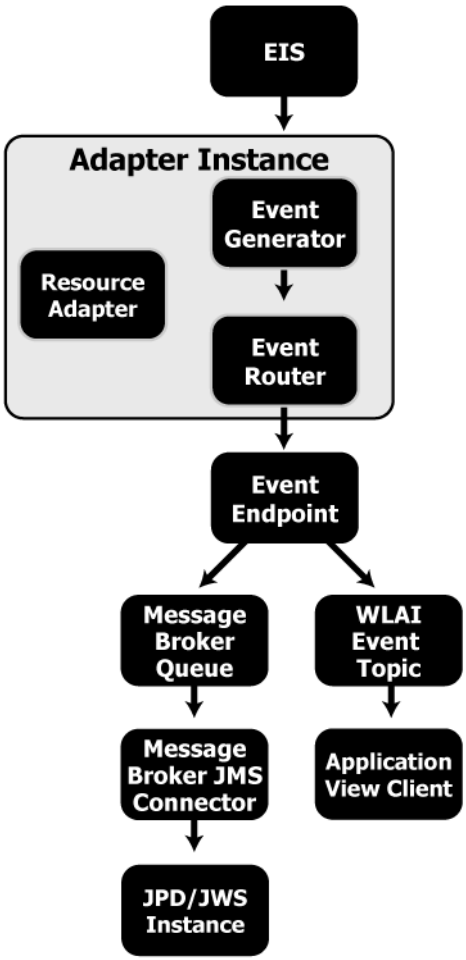
This section walks through the process of an event notification at run time. Event notifications are always asynchronous and are published to two destinations:

- **Message Broker channel**. For business processes (JDP files) or web services (JWS files) defined in WebLogic Workshop. Clients are subscribed to listen for events on specific channels in the Message Broker queue. The Message Broker uses channel files that describe the type of data being published to any given Message Broker channel. The channel file associates a channel name with a schema definition. At design time, when defining the application view, the project associated with the event must be specified. Thereafter, when an application view is deployed, a channel file containing event type information (comprised of the application name, application view name, and event name) is saved in the WebLogic Workshop application directories.
- **Application Integration event topic**. (`WLAI_EVENT_TOPIC` JMS topic). A message-driven bean (the WLI-AI Event Processor) listens on the `WLAI_EVENT_QUEUE` distributed destination and publishes a copy of the event to the `WLAI_EVENT_TOPIC`.
  The `WLAI_EVENT_TOPIC` is a distributed JMS topic that handles the delivery of events to remote application view clients.

At design-time, event consumers must be configured to listen for the event on either of these destinations. In addition, the EIS must be configured to send event messages to a particular destination so that the adapter can receive it via the EIS-specific communications protocol.

The following figure shows how an event notification is processed at run time.

Figure 2-4 Run-Time Processing of an Event Notification



The following procedure describes, at a high level, how an event notification is processed at run time:

1. The adapter receives the event message from the EIS.
2. The adapter translates the event message to the appropriate XML format using the event schema that was configured for the event.

3. The Event Generator receives the inbound message and then posts it to the Event Router.
4. The Event Router forwards the event message to the Event Endpoint.
5. The Event Endpoint sends the event XML document to two destinations:
   - **WebLogic Integration Message Broker** for delivery to any subscribers to the Message Broker channel. Subscribers use either a Message Broker Subscription control or, for business processes only, a Message Broker static subscription.
   - **Application Integration event topic** (`WLAI_EVENT_TOPIC`) for delivery to remote or local Application Integration clients who are listening to this topic.
6. Event consumers who have subscribed to the event receive the event XML document and process it accordingly.

CHAPTER 3

Roles, Responsibilities, and Tasks

This section provides an overview of the roles and tasks required to create integration solutions. This section includes the following topics:

- Roles and Responsibilities
- Process for Creating Integration Solutions
- Where To Go From Here

Roles and Responsibilities

The following sections describe the roles that must be fulfilled for members of an integration solution team:

- Application Integration Specialists
- EIS Specialists
- Technology Specialists

A successful integration solution requires input from all of these participants. Depending on the solution, one person may assume multiple roles and all roles might not be required.

Application Integration Specialists

Application integration specialists lead the implementation of an integration solution and drive the design effort. Application integration specialists are knowledgeable about the features and capabilities of the WebLogic Integration product, particularly the application integration

capabilities. They consult with EIS specialists to determine requirements, map those requirements to WebLogic Integration features, and design an integration solution's architecture. Integration specialists are responsible for the end-to-end solution and have experience in the following areas:

- Business and technical analysis
- Architecture design
- Project management

EIS Specialists

EIS specialists are experts in the enterprise information systems (EIS) that are part of the integration solution. An EIS specialist provides the information needed to integrate the EIS into the integration solution, including external interfaces, connection protocols, EIS metadata and data formats, and EIS behaviors. EIS specialists are knowledgeable about all aspects of the applicable EIS system and they have experience in the following areas:

- Technical analysis
- Integration solution design
- In-depth knowledge of the organization's EIS deployment and operations

Technology Specialists

Technology specialists are experts in the various technologies used in integration solutions. Examples of technology specialists include:

- Java developers
- database administrators
- system administrators
- infrastructure specialists, such as experts in network, intranet, extranet, and mail infrastructure

Process for Creating Integration Solutions

This section provides a high level, end-to-end view of the process of creating integration solutions that involve EIS integration. It includes the following topics:

- Phase 1: Design the Solution
- Phase 2: Build the Solution
- Phase 3: Deploy and Manage the Solution

This section is a hypothetical or idealized solution designed to showcase product features rather than a suggestion of how to execute a plan. It is intended to supplement any methodologies or processes already used in your organization to build and deploy integration solutions.

Phase 1: Design the Solution

This phase involves two steps: defining the components of the integration solution and creating a detailed, end-to-end design.

Step 1: Define the Components of the Solution

The first phase is to define the components of an integration solution, which includes (but is not limited to) the following tasks:

- Determine which business process(es) will be involved in the integration solution.
- Determine which external EISs and other technologies will be involved in the integration, as well as any external EIS interfaces involved in the business process(es) that you are integrating.
- Determine which WebLogic Platform components will be involved in the integration solution, such as:
  - web services, business processes, or portals designed in WebLogic Workshop
  - custom applications
- Determine which adapters will be required, including BEA WebLogic Adapters for WebLogic Integration and, if applicable, custom adapters. An integration solution can involve multiple adapters.

For more information, see "Understanding Application Integration" on page 2-1.

Step 2: Create a Detailed, End-to-End Design of the Solution

Once you have defined the components of the solution, you need to create a detailed design that specifies:

- Any service invocations, including:
  - the schemas required for requests and responses (as described in "EIS Metadata, Schemas, and Repositories" on page 2-11)
  - the client application that will initiate each service request and handle the response (as described in "Clients for Service Invocations" on page 2-7)
  - whether each service invocation will be synchronous or asynchronous
  - other requirements that pertain to the associated EISs
- Any event notifications, including:
  - the event schema required (as described in "EIS Metadata, Schemas, and Repositories" on page 2-11)
  - the event consumer(s) that will subscribe to, and listen for, events initiated by the EIS, as described in "Event Consumers" on page 2-8
  - the configuration required on each EIS to publish events to destination where the adapters can receive them
  - other requirements that pertain to the associated EISs
- Any requirements for connecting to the EIS, such as login credentials, network connections, specialized configuration, and so on.
- Any specialized business logic, such as transaction processing.
- Any other components of the integration solution, such as business processes, web services, portals, and so on.

This step involves the expertise of business analysts, system integrators, and EIS specialists. Note that an integration solution can be part of a larger integration solution.

Phase 2: Build the Solution

The next phase is to build the solution using the design-time tools described in "Tools for Integration Solutions" on page 2-13. Build tasks include:

- Purchase, install, and configure the WebLogic Platform and any adapters.
- Create the schemas for services and events according to the "Generating Schemas" chapter in your adapter documentation, using the BEA Application Explorer if appropriate for your adapter.
- Create the application views that provide an XML-based interface between WebLogic Server and the EIS. For each application view, you configure connection information, services, and events.
- Build and integrate with other BEA software components as required. For example, you might need to build business processes, web services, or portals in WebLogic Workshop and configure them to invoke services or receive and process event notifications. Similarly, you might need to construct queries in Liquid Data for WebLogic that access application views as data sources. For instructions, see the documentation associated with the BEA software component you are using.
- Test the end-to-end solution, making sure that all of its components interact correctly and produce the desired results.

This phase involves the expertise of technical specialists, such as designers (of business processes, web services, portals, and queries), developers, system integrators, database administrators, EIS specialists, and so on.

Phase 3: Deploy and Manage the Solution

The final phase is to deploy the integration solution in a production environment and monitor its ongoing operation.

- Design the deployment.
- Deploy the required components of the BEA WebLogic Platform.
- Install and deploy the adapter as described in the adapter's *Installation and Configuration Guide*.
- Deploy any application views and schemas for EIS integration.
- Verify business processes in the production environment.

- Monitor, tune, and troubleshoot the deployment.

This phase requires system administrators, network administrators, network operators, and specialists who operate the infrastructure of your organization.

Where To Go From Here

To begin using one of the BEA WebLogic Adapters for WebLogic Integration in your integration solution, refer to the "Getting Started" section of the Introduction in the adapter's *User's Guide*.

CHAPTER 4

Understanding the ADK

If you are an adapter provider or developer, you can use the WebLogic Integration Adapter Development Kit (ADK) to create your own J2EE-compliant EIS adapters. The ADK is a collection of four frameworks, each of which comprises tools and Java classes. Together, these frameworks let you quickly develop adapters that you can easily test, package, and distribute. For details about using the ADK frameworks to create adapters, see *Developing Adapters*.

This section provides information about the following four component frameworks:

- Design-Time Framework
- Run-Time Framework
- Logging and Auditing Framework
- Packaging Framework

Design-Time Framework

An adapter's design-time interface lets you define and deploy application views on the instance of WebLogic Server that is hosting WebLogic Integration. Whenever you build an adapter with the WebLogic Integration ADK, you can also develop a design-time user interface for it. The design-time user interface for an adapter can be accessed from any common Web browser. The interface allows users who are not programmers to interact with the adapter without writing code. For example, by using a user interface, business analysts can log in to an adapter and define their own custom application views. Without a design-time user interface, an adapter can be used only by highly technical users.

To facilitate the development of design-time user interfaces for adapters, the ADK includes a set of Java classes and tools known as the *design-time framework*. This framework is a powerful feature because it allows users who are not programmers to use an adapter. By simplifying the development process, the ADK's design-time framework expands the audience for the adapter and upgrades the role played by business analysts in the implementation of business processes.

Run-Time Framework

The ADK provides a *run-time framework*: a complete, extensible event generator that supports the development of events. To help you develop services, the run-time framework provides a J2EE-compliant adapter that offers a complete set of the minimum functions. Adapter developers can save coding and debugging time by starting with this base framework and extending it to meet the needs of their enterprise.

Logging and Auditing Framework

If you are an administrator, it is essential to create adapters that automatically log alert messages on your system that can be audited later. To make it easy to develop an adapter with built-in logging and auditing support, the ADK includes a logging and auditing framework. Any adapter you develop can generate internationalized and localized alert messages and can deliver these messages to multiple output destinations.

Packaging Framework

If you are a third-party adapter provider, you can use the ADK packaging framework when preparing your adapter for delivery to a customer. This framework makes it easy to create the archive and environment files required for packaging.

CHAPTER 5

Understanding the Development Kit Adapters

The BEA WebLogic Integration Adapter Development Kit (ADK) provides sample adapters to get you started developing your own adapters: two DBMS adapters and a sample adapter. If you are developing your own adapters using the ADK, we recommend that you begin by studying the adapters supplied with the kit. Although these adapters are generic and simple, they serve as excellent examples of the types of adapters you can build using the ADK.

How the Kit Adapters Were Developed

All the development kit adapters were developed using the ADK. Although the ADK makes it possible to develop sophisticated adapters, the adapters provided in the kit have been kept simple deliberately, to make them easy to dissect and understand.

How to Use the Kit Adapters

If you are an adapter provider or developer, we recommend that you study the kit adapters to increase your knowledge of the ADK and to determine how you can use the adapters as models for your own. All kit adapters are based on a superset of J2EE Connector Architecture 1.0 (J2EE 1.3 from Sun Microsystems). XML input/output and browsing requirements have been added to the CCI interface (collectively known as XCCI for XML CCI). For details about the ADK, see *Developing Adapters*. The following sections provide details about the kit adapters.

DBMS Adapter

WebLogic Integration provides DBMS adapters that integrates WebLogic Server with a simple relational database that it uses as its EIS. Two DBMS sample adapters are provided:

- BEA_WLS_DBMS_ADK—A sample DBMS adapter that includes XA transaction support. This adapter is used for the tour of the sample adapter and the description of adapter development.
- BEA_WLS_DBMS_ADK_LOCALTX—A sample DBMS adapter that includes only support for local transactions.

Warning: When setting connection parameters for these adapters, specify a JDBC Driver and URL. Do not use a DataSource with the sample DBMS adapters. Using a DataSource with the sample adapters results in `java.sql.SQLException XA` errors.

The DBMS adapters serve as a good example for adapter developers and providers who want to understand the adapter and ADK without having to learn an unfamiliar proprietary database system.

If you do not have a suitable database to use with the DBMS adapters, you can use the PointBase database included with WebLogic Integration.

The DBMS adapters support the following functions:

- Retrieving all database records
- Retrieving sets of database records
- Writing database records
- Receiving notifications when a record is updated, inserted, or deleted from a table

For more information, see "Learning to Develop Adapters Using the DBMS Sample Adapters" in *Developing Adapters.*

Sample Adapter

The sample adapter is provided as a template for new adapters. It includes a design-time component, a service connection, and an event connection. The source code clearly documents the structure of the service connection and the event connection.

The sample adapter is accompanied by a simple EIS implementation that demonstrates how related events can be triggered by invoking services. Each component contains comments indicating where you must supply your own adapter-specific logic.

Index