BEA WebLogic Integration™

Designing BEA WebLogic Integration Solutions

Release 7.0
Document Date: June 2002

Designing BEA WebLogic Integration Solutions

Part Number	Date	Software Version
N/A	June 2002	7.0

Contents

About This Document

This document describes how to design an integration solution for the BEA WebLogic Integration environment. It defines key design concepts, provides a roadmap for determining integration requirements based on a comprehensive analysis of business and technical requirements, and describes how to design an integration solution that meets goals for high availability, reliability, performance, scalability, and security.

Overview Documents for WebLogic Integration

This document is one in a series of four documents that provide an overview of BEA WebLogic Integration, and that explain how the functionality provided by WebLogic Integration is used at various stages in the design, development, and deployment of integrated solutions. Readers should start with these documents to gain a comprehensive understanding of the functionality provided by WebLogic Integration. The other documents in the series are:

- *Introducing BEA WebLogic Integration*—Provides an overview of WebLogic Integration. It outlines the integration problems today's e-businesses face, with their collections of fragmented, heterogeneous business systems. It also describes the application integration, B2B integration, business process management, and data integration functionality WebLogic Integration provides to solve e-business integration problems.
- *Learning to Use BEA WebLogic Integration*—Describes a sample integrated application. The sample application deploys a supply chain hub, which connects with business partners, automates a number of business processes, and integrates back-end enterprise information systems. Readers learn how to set up and run

the sample application, and understand how the integrated solution is architected and developed using WebLogic Integration.

- *Deploying BEA WebLogic Integration*—Explains how to move an integrated solution from a development to a production environment. Readers learn about configuring, scaling, porting, and performance tuning integrated applications.

Once you are familiar with the contents of these overview documents, you can proceed to the detailed documentation about the functionality provided by WebLogic Integration.

What Is In This Document

This document is organized as follows:

- Chapter 1, "Introduction," introduces key design concepts, roles, and design tasks.
- Chapter 2, "Determining Integration Solution Requirements," provides a roadmap for determining integration requirements based on business and technical analysis.
- Chapter 3, "Designing the Integration Solution," provides a roadmap for designing the architecture of a solution that meets the integration requirements in the WebLogic Integration environment.

What You Need to Know

This document is intended primarily for integration specialists who are responsible for researching and designing integrations that will be implemented using WebLogic Integration.

Integration specialists work with business analysts and technical analysts to define the goals, requirements, and scope of each integration project. Integration specialists then design an integration architecture for the WebLogic Integration environment based on design patterns and best practices.

Integration specialists must therefore have a thorough understanding of the WebLogic Integration architecture and capabilities. Before using this document, integration specialists should begin by reading *Introducing BEA WebLogic Integration*.

This document refers frequently to the WebLogic Integration sample application. Before using this document, integration specialists should become very familiar with this sample application by reading *Learning to Use BEA WebLogic Integration*, particularly Chapter 1, “Introduction,”and Chapter 3, “Designing the Integration Solution.”

How to Print this Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the WebLogic Integration documentation CD. You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format.

If you do not have the Adobe Acrobat Reader installed, you can download it for free from the Adobe Web site at `http://www.adobe.com/`.

Contact Us!

Your feedback on the WebLogic Integration documentation is important to us. Send us e-mail at **docsupport@bea.com** if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the WebLogic Integration documentation.

In your e-mail message, please indicate that you are using the documentation for BEA WebLogic Integration Release 7.0.

If you have any questions about this version of BEA WebLogic Integration, or if you have problems installing and running BEA WebLogic Integration, contact BEA Customer Support through BEA WebSUPPORT at **www.bea.com**. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

Convention	Item

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

|	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

Convention	Item

`...`	Indicates one of the following in a command line:
	■ That an argument can be repeated several times in a command line
	■ That the statement omits additional optional arguments
	■ That you can enter additional parameters, values, or other information
	The ellipsis itself should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 Introduction

The following sections introduce the process of designing an integration solution:

- Key Integration Design Concepts
- Roles in Integration Solution Design
- Key Design Tasks

The purpose of this document is to provide an integration specialist with a roadmap for designing solutions built on WebLogic Integration. This document describes key WebLogic Integration issues that an integration specialist needs to identify at the business and technical requirements level and then describes how to implement those key concepts in an integration solution architecture design using WebLogic Integration.

Note: A total integration solution involves many methodologies that are outside the scope of this document, such as integration solution methodologies, project management methodologies, business and technical specification methodologies, and so on. This document supplements those methodologies by addressing the portion of the total integration solution that is directly related to designing an integration solution on WebLogic Integration.

This document refers extensively to the sample WebLogic Integration application for examples of how to determine business requirements and design an integration. Integration specialists should familiarize themselves with this sample application by reading *Learning to Use BEA WebLogic Integration*.

Key Integration Design Concepts

The following sections describe key design concepts:

- Business Processes
- Integration Solutions
- Design Goals
- Design Considerations
- Best Practices and Design Patterns

Business Processes

A *business process* is a set of interconnected business activities with a beginning and an end, and clearly defined inputs and outputs. A business process includes all the information associated with a particular integration solution: an initiating event, participating applications, event definitions, data transformation, mapping, process definitions, and so on. Business processes can be simple, such as obtaining the shipping address from a sales order, or more complex, such as asking multiple suppliers to provide price and availability information for a particular item.

Integration Solutions

An *integration solution* involves the integration of one or more related business processes. For example, a company might want to better integrate its purchasing operations with suppliers, and the integration solution could include two business processes:

- Getting bids from suppliers
- Processing purchase orders

A business process might be broken down into component pieces, such as:

- Create and approve a purchase order
- Update or edit a purchase order
- Process change orders for a purchase order
- Cancel a purchase order

An integration solution can span multiple system, geographic, and organizational boundaries. For example, getting bids from different suppliers over the Web involves integration of the buyer's purchasing system and the suppliers' sales and inventory systems.

For more information about integration solution, see "Defining Integration Solutions and Business Processes" on page 2-2 and "Specifying the Integration Solution and Business Processes" on page 2-17.

Design Goals

Design goals are the broad criteria by which the success of an integration solution is measured. Common design goals include:

- Availability—An integration solution must be sufficiently available and accessible, with fail-over provisions in the event of hardware or network failures.
- Reliability—An integration solution must reliably manage transactions, process and validate data, and ensure data integrity.
- Performance—An integration solution must deliver sufficient performance at peak and off-peak loads.
- Scalability—An integration solution must be sufficiently scalable to handle anticipated increases in loads.
- Security—An integration solution must sufficiently protect data from unauthorized access or tampering.

For each integration solution, design goals are identified and prioritized. WebLogic Integration provides all the features needed to meet these design goals.

Design Considerations

Design considerations are issues that need to be evaluated when designing an integration solution. For example, qualities of service—performance, availability, reliability, scalability, response times, security, logging, and auditing—must be defined, analyzed, and factored into any integration solution design. This document identifies the design considerations that are relevant to almost any integration solution built with WebLogic Integration.

Best Practices and Design Patterns

Best practices are methods that are proven to work most effectively in certain situations. For example, before attempting to design an integration solution, as described in Chapter 3, "Designing the Integration Solution," it is best to perform due diligence by thoroughly defining the business and technical requirements, as described in Chapter 2, "Determining Integration Solution Requirements." The goal of this document is to describe best practices for WebLogic Integration, supplementing an organization's integration solution design methodology with practical ideas that reflect the collective experience of those who have successfully implemented WebLogic Integration in a variety of organizations.

Design patterns are recommendations about how to best design all or part of an integration solution to support a specific business requirement such as high performance, data integrity, and so on. For example, if a BPM workflow uses an invoke service activity to call an external service such as a business operation (Java class or EJB), it should invoke only a single service. Multiple service invocations should be implemented using multiple Task nodes—one Task node per service invocation. For more information about Task nodes, see "Placing Actions in Task Nodes" in "Understanding Actions" in "Defining Actions" in *Using the WebLogic Integration Studio*.

Roles in Integration Solution Design

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

- Integration Specialists
- Business Analysts
- Architects
- Enterprise Information System Specialists
- System Administrators

A successful integration solution design requires input from *all* of these participants. Note that one person can assume multiple roles, and not all roles are relevant in all integration solutions.

Integration Specialists

Integration specialists lead the implementation of a WebLogic Integration solution and drive the design effort. Integration specialists are knowledgeable about the features and capabilities of the WebLogic Integration product. They consult with business and architects to determine requirements, map those requirements to WebLogic Integration features, and design the architecture of an integration solution. Integration specialists have experience in the following areas:

- Business and technical analysis
- Architecture design
- Project management

Business Analysts

Business analysts provide expertise in an organization’s business processes, procedures, policies, business rules, and resources. They are knowledgeable about operations and have experience in the following areas:

- Business analysis
- Process design and modeling

Architects

Architects are technical specialists that provide expertise in an organization’s information technology infrastructure, including telecommunications, platforms, applications, data repositories, future technologies, and IT organizations. They are knowledgeable about information systems and have experience in the following areas:

- Technical analysis
- System architecture
- Application design

Enterprise Information System Specialists

EIS specialists are experts in the enterprise information systems (EIS) that are being integrated. An EIS specialist provides the information needed to connect the EIS into the integration solution. These systems are usually connected to WebLogic Integration through WebLogic Integration adapters. EIS specialists are knowledgeable about all aspects of the EIS system being integrated (including the data formats, behavior, and external interfaces), and they have experience in the following areas:

- Technical analysis
- Application integration solution design

System Administrators

System administrators provide in-depth technical and operational knowledge about databases and applications that are deployed in an organization. They are knowledgeable about deployment topologies and have experience in the following areas:

- Capacity and load analysis
- Performance analysis and tuning
- Support planning

Key Design Tasks

To design an integration solution, the integration solution design team must perform the following basic tasks:

1. Define the integration solution requirements, based on a thorough analysis of business and technical requirements. For more information, see Chapter 2, "Determining Integration Solution Requirements."
2. Design the integration solution architecture, based on a thorough analysis of business and technical requirements plus an understanding of the WebLogic Integration features that are best suited to meet those requirements. For more information, see Chapter 3, "Designing the Integration Solution."

After completing the design for an integration solution, the integration specialist works with programmers, EIS specialists, and system administrators to construct the WebLogic Integration solution. For more information, see "Design & Deploy" in the WebLogic Integration product documentation Web site.

CHAPTER

2 Determining Integration Solution Requirements

Note: The cXML business protocol is deprecated as of this release of WebLogic Integration. For information about the features that are replacing it, see the *BEA WebLogic Integration Release Notes*.

An integration specialist must investigate the business and technical requirements for an integration solution. The integration specialist works collaboratively with business analysts and architects to clarify these requirements and to achieve a consensus on the organization's needs and priorities. Business analysts and architects provide the detailed requirements definitions that an integration specialist uses to design an integration solution architecture on WebLogic Integration.

With one exception, the following sections describe issues to consider when determining the requirements of an integration solution. The final section shows how these issues relate to the sample WebLogic Integration application.

- Defining Integration Solutions and Business Processes
- Defining Actors and Their Roles in Business Processes
- Defining Business Events
- Defining Business Data Flows
- Defining the Quality of Service
- Defining the Integration Solution Topology

- Specifying Requirements for the Sample WebLogic Integration Application

Defining Integration Solutions and Business Processes

When determining integration solution requirements, the integration specialist must:

- Clarify the scope of the integration solution and then divide the work into a collection of smaller, discrete units of work. An *integration solution* consists of one or more related business processes, as described in “Integration Solutions” on page 1-2.
- Identify the business processes that are candidates for an integration solution. A *business process* is a set of interconnected business activities with a beginning and an end, and clearly defined inputs and outputs, as described in “Business Processes” on page 1-2.

To see how these tasks apply to the sample WebLogic Integration application, see “Specifying the Integration Solution and Business Processes” on page 2-17.

Each business process should consist of a single process definition that describes the required business activities. Integration specialists, business analysts, and architects work together to define each business process. The integration specialist must ask questions such as the following:

- What work must be done for each business process?
- What are the principal activities for each business process?
- What are the characteristics for each business process or component?
  - Sequence or order (in relation to other tasks and activities)
  - Frequency of occurrence
  - Urgency and timeliness
  - Duration or elapsed time
  - Volume range, including average volume, peaks, and lulls

To see how this task applies to the sample WebLogic Integration application, see "Specifying the Integration Solution and Business Processes" on page 2-17.

Defining Actors and Their Roles in Business Processes

Actors are people and processes that produce and consume business events. A business process defines the relationships among the actors, the roles of the actors in the process, and the business events that are exchanged by the process and the actors. The integration specialist identifies the actors and their roles, as described in the following sections:

- Analyzing Actors
- Analyzing Roles
- Classifying Actors by Type

To see how this task applies to the sample WebLogic Integration application, see "Specifying Actors and Roles" on page 2-18.

Analyzing Actors

For each business process, the integration specialist asks questions such as the following:

- What are the actors of each business process? Who are the players? Do they include, for example, other departments within a company, vendors, customers, systems, and Web users?
- What is the geographic topology of these entities? Are they all located at one site, or are they distributed across other locations? Which entities are located at which site?
- How is data transported between actors?

  - At a single location (within a LAN or domain)?
  - Across a WAN to other locations within the same organization?
  - Outside the WAN to actors beyond a firewall?
- What is the format of the events flowing in and out of each entity in each business process?

Analyzing Roles

To define the role of each entity in the business process, the integration specialist must answer questions such as the following:

- What roles are involved in the business process?
- What is the sequence in which the work of each role is performed?
- How does the work of each role fit into the overall business process? Specifically, does the player of a given role:
  - Initiate the process?
  - Provide a service to the process?
  - Consume business events generated by the process without returning a response?
  - Review, approve, reject, or otherwise make a decision along the way?
- How many people and processes are associated with each role?

Classifying Actors by Type

The integration specialist must identify and characterize the different types of actors involved in a business process. The actors in a business process include both humans, who use the WebLogic Integration Worklist and other client applications, and various software entities.

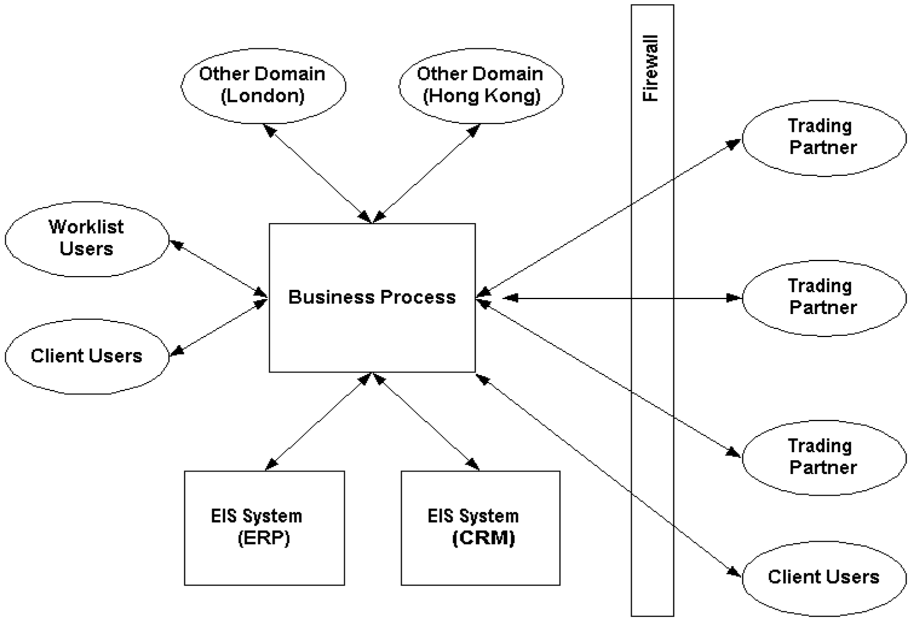
Note: The Worklist client is deprecated as of this release of WebLogic Integration. For information about the features that are replacing it, see the *BEA WebLogic Integration Release Notes*.

The following sections describe several types of actors:

- Human Users
- Application Integration—Enterprise Information Systems (EIS) Applications Used for Application Integration
- Other Domains in the Enterprise
- External Trading Partners Performing B2B Integration

The following illustration shows these actors:

Figure 2-1 Actors in a Business Process



Human Users

Human users fall into two categories:

- *Worklist users* are assigned work directly by the business process. For example, a manager using the Worklist might be assigned the task of approving all orders over $100,000. Users can be grouped by role. Tasks can be assigned either to individual users or to users associated with a role. Users who are assigned tasks in this way work closely with a business process.

  Note: The Worklist client is deprecated as of this release of WebLogic Integration. For information about the features that are replacing it, see the *BEA WebLogic Integration Release Notes*.

- *Client users* interact with a business process indirectly, by using applications that are part of the process. For example, this type of user might initiate a *create order* process from a Web-based GUI and expect to receive confirmation of an order via e-mail.

The integration specialist must identify the type of client required by each user, such as a Swing GUI, HTML, e-mail client, and so on.

Application Integration—Enterprise Information Systems (EIS) Applications Used for Application Integration

The technical infrastructure of an organization can consist of a variety of enterprise information systems (EIS), including legacy mainframe systems, client-server applications, and packaged applications, such as ERP and CRM applications.

The integration specialist focuses on EIS applications that are directly involved in the integration solution, asking questions such as the following:

- Will the integration solution have a direct connection to the EIS?
- Can the EIS be customized to support the integration?
- Will the EIS and the business process exchange events?

The integration specialist needs to identify the EIS applications that are involved in the business processes and for which the answer to these questions is yes.

In addition, the integration specialist must determine the following:

- For each EIS application, the business functionality that it provides (such as purchase order management, inventory management, and so on)
- For each EIS application that participates in the business process, the application details and interfacing technology that will be used

Examples of EIS Applications

Examples of EIS applications include:

- Packaged applications (including e-mail)
- Custom applications (WebLogic Server and nonWebLogic Server applications)
- Applications that use middleware technologies (such as CORBA or MQSeries)
- Applications with queue-based interfaces (such as MQSeries)
- Technology adapters (file or database adapters)

Analyzing EIS Integration Requirements

The integration specialist evaluates application requirements, asking questions such as the following:

- What are the name and version of the EIS?
- On which platform(s) does it run?
- What interfacing technology does the EIS offer? Does the EIS offer an API? Does the EIS have a file-based interface? Does the EIS use a middleware-based interface?
- Do the EIS internal business processes that will be involved in the integration mandate the use of a particular EIS interface?
- What is the format of the data passed by the interface (binary, XML, other)? Is metadata (data that describes the data) available, or does this metadata need to be created by hand?
- For each business event that is exchanged between the business process and the EIS application, what is the EIS interface (API, event, database table, and so on) that will be used?

- What are the human and application interfaces?
- What are the mappings between interfaces?

Determining Whether Additional Customization is Required

The EIS applications might not implement all the functionality required to support the integration solution and might therefore require internal customization. The integration specialist needs to determine whether this is needed and, if so, define the technical requirements.

Other Domains in the Enterprise

An integration solution might involve contact with other locally-administered domains in the enterprise, such as remote offices. In such situations, the internal mechanics of the other domain are beyond the control of the integration solution. Specifically, the other domain might or might not use WebLogic Integration as its integration solution.

The *interface* between the business process and other domains is the set of business events that flow between them. This interface must be defined.

One approach is to represent independent domains as trading partners *within* the enterprise, using B2B integration functionality to support the flow of information among them. For example, suppose an enterprise owns a factory in Malaysia that supplies parts to a wholly-owned assembly-line factory in Japan. Using B2B integration, the Malaysian office can act as a supplier (“seller”) of components to the Japanese office, which acts as a buyer.

External Trading Partners Performing B2B Integration

An integration solution might involve external trading partners with which a business process needs to communicate. Such processes cross the enterprise firewall. For trading partner integration, the integration specialist must define:

- What is the nature of the B2B integration?
  - Is it a value-chain integration? If so, is it a supply-chain integration (where the trading partners are suppliers to the enterprise), a demand-chain integration (where the trading partners are customers to the enterprise), or a combination of both?
  - Is it a B2B exchange that can enlist trading partners for B2B transactions?

- Are there mandated protocols that must be included in the integration solution (XOCP, RosettaNet, cXML, or EDI)?

  Note: The XOCP and cXML business protocols are deprecated as of this release of WebLogic Integration. For information about the features that are replacing them, see the *BEA WebLogic Integration Release Notes*.

Defining Business Events

The following sections describe how to define business events:

- About Business Events
- Defining Business Event Characteristics
- Defining the Message Format for Business Events
- Specifying Business Events

To see how this task applies to the sample WebLogic Integration application, see “Specifying Business Events” on page 2-20.

About Business Events

Business events are exchanges of messages or tasks that occur between actors during a business process. A business event indicates that a business activity has taken place and needs to be performed. For example, an EIS could publish a *new customer created* event, or an order management application could subscribe to *new order* events to process new orders. Each actor can exchange business events with the business process. Each business event should be given a descriptive name that uniquely identifies it in the business process.

Business events contain business data. For example, suppose a *create new customer* event occurs during the execution of a customer management application. In the process of creating a new customer, the application might receive the name and address of the customer name from another actor in the business process and return a number for the customer to that actor.

Defining Business Event Characteristics

For each interaction between an actor and the process, the integration specialist must define:

- Business events that can be sent and received
- Any unique characteristics of elements in the business events

Defining the Message Format for Business Events

In WebLogic Integration, business events are sent as either XML or binary messages. The integration specialist must define:

- Business event format (XML or binary) and metadata. A common format (such as XML) is used whenever possible. For application integration with EISs, the adapter converts proprietary EIS protocols or message formats to the common format.
- Any information in the event that may be used for controlling decisions (such as conditional processing, routing, triggering, and so on)

Defining Business Data Flows

The business process defines the required flow of data between actors. From this flow, the integration specialist can determine how the business data needs to be manipulated. Perhaps, for example, the business data should be split into multiple events, concatenated from multiple events, or transformed from one data format to another.

This data flow should also define any business data that is used in the business process to make processing decisions. For example, if the business process includes an algorithm specifying that purchase orders over $5,000 must be approved by a vice president, then the amount of money in each purchase order should be defined as required business data.

The following sections describe the work of preparing data flow requirements:

- Defining Data Flow Requirements
- Analyzing the Data Flow
- Specifying Data Flow Requirements

To see how this task applies to the sample WebLogic Integration application, see "Specifying Data Flow Requirements" on page 2-20.

Defining Data Flow Requirements

For each data flow, the integration specialist needs to define the following requirements:

- *Conditional data*—data that is required to make processing decisions. This data is extracted from business events.
- *Business or application rules*—processing rules that are applied to the conditional data to determine the run-time execution path of the process.
- *Mappings*—data transformations between the business events used as input and those used as output.
- *Business transactions*—transactional boundaries in the process. A single process might contain many business transactions. In addition, for each business transaction, the integration specialist needs to define any compensating actions that must be performed if a transaction needs to be rolled back.
- *Error handling*—what exceptions can occur and how they should be handled.

Analyzing the Data Flow

The integration specialist needs to analyze the technical aspects of the data flow, asking questions such as the following:

- What are the characteristics of each data element?
- What are the characteristics of messages?
  - Message size, specified in terms of minimum, maximum, and average size

  - Message volume, specified in the number of messages at peak, lull, and average volumes, plus any cyclical patterns
  - Single or batched (aggregated) message. If messages are aggregated, do they need to be split up and routed appropriately? If so, what are the routing criteria or conditions?

- What data transformations are needed between the source data and target data?

For example, suppose an order management application sends a new order event to a shipping application for processing. Suppose that a shipping application runs in three separate regional offices (eastern, central and western) as three separate instances. The order management application needs to notify the appropriate application instance based on the ship to address in the order. In addition, the order management application needs to notify the billing application of the new order.

In this scenario, the data flow requirements would be:

- Three business events:
  - *New order* event from the order management application
  - *Ship goods* event to the shipping application
  - *Send invoice* event to the billing application
- Conditional data—the state to which the order will be shipped. This information is extracted from the billing address in the new order event.
- Business rules—the rule used to identify the shipping application instance to receive the *ship goods* event, which is based on the ship to state/province in the order and the list of states/provinces associated with each application instance.
- Mappings—the data transformation mappings from:
  - *New order* event to the *ship goods* event
  - *New order* event to the *send invoice* event
- Business transaction—Either both the shipping and billing applications are updated successfully, or neither is updated. Compensating actions for rolling back each application if the other fails must be defined.
- Error handling—if a data error occurs, such as an order is received with an invalid state/province.

Defining the Quality of Service

The integration specialist must define the following Quality of Service characteristics in business and technical terms. The following sections describe these characteristics:

- Performance
- Availability and Reliability
- Response Times
- Security
- Scalability
- Logging and Nonrepudiation

Performance

The integration specialist must answer questions such as the following:

- How quickly, in business terms, must the business process be carried out?
- What are peak and off-peak performance requirements?

For example, an integration specialist might specify that the system must be capable of handling 20,000 orders per business day (between 9am and 5pm) with a maximum load of 5,000 orders at any one time.

For more information about WebLogic Integration performance, see "Performance Considerations in Integration Design" on page 3-28. To meet performance requirements, high-volume integration solutions might require a clustered configuration. For more information about clustering, see "Understanding WebLogic Integration Clusters" in *Deploying BEA WebLogic Integration*.

Availability and Reliability

The integration specialist must define the availability requirements for the business process, asking questions such as the following:

- When must systems be available? Are they needed 24 hours a day, 7 days a week (24x7)?
- What are the planned and anticipated periods of scheduled and unscheduled system downtime, respectively?
- What is the maximum allowable downtime?
- What failover and recovery protections are required in case a hardware or network failure occurs?
- Do business messages need to be persisted while in transit and recovered upon a system restart?

Some systems might require 24x7 availability with instant failover, while others might require availability only during the business day with scheduled maintenance on evenings or weekends. To meet reliability requirements, high-availability integration solutions require a clustered configuration, which is described in "Configuring a Clustered Deployment" in *Deploying BEA WebLogic Integration*.

Response Times

The integration specialist must define the response time requirements for the business process. For example, suppliers might need to respond to a bid request within 24 hours, or suppliers might need to be notified within 60 minutes of the bid award. An event might time out if the response time limit has elapsed.

Security

The integration specialist must define the security requirements for the business process, asking questions such as the following:

- How sensitive is the information in the business process?

- What are the privacy needs associated with each role?
- What security safeguards are currently in place?

Scalability

The integration specialist must define the scalability requirements for the business process, based on the current volume of work and the projected volume in the future. For example, an order-processing integration solution might need to be able to handle triple, within two months, the volume of orders it can handle, without interruption to service or additional application development. To meet scalability requirements, integration solutions can use a clustered configuration, which is described in "Configuring a Clustered Deployment" in *Deploying BEA WebLogic Integration*.

Logging and Nonrepudiation

The integration specialist must define the system logging requirements for the business process, asking questions such as the following:

- What kinds of problems can arise?
- What information needs to be logged and monitored?
- For integration solutions that use B2B integration, what information needs to be logged and maintained for audit or nonrepudiation purposes?

  Nonrepudiation is the ability of a trading partner to prove or disprove having previously sent or received a particular business message to or from another trading partner. Nonrepudiation of origin and nonrepudiation of receipt might be required by law for critical business messages.

Defining the Integration Solution Topology

The integration specialist must define the integration topology that specifies the physical location of the various entities in the integration environment. This topology should include information about the types of network connections among the participants, including LAN, WAN, Internet, dial-up, and so on.

The integration specialist must ask whether all the constituent entities in an integration solution will be located:

- At a single physical site (a LAN with a single domain)?
- At different sites that are connected via a WAN behind a firewall?
- At different sites for various trading partners, separated by firewalls?

Specifying Requirements for the Sample WebLogic Integration Application

The following sections explain how the requirements are defined for the sample WebLogic Integration application:

- Specifying the Integration Solution and Business Processes
- Specifying Actors and Roles
- Specifying Business Events
- Specifying Data Flow Requirements
- Specifying the Quality of Service
- Specifying the Integration Topology

Specifying the Integration Solution and Business Processes

In the sample WebLogic Integration application, General Control Systems (GCS) defines a single integration solution, purchasing integration, to better manage the value chain with its suppliers. Their analysis reveals that they need to define integration requirements for two business processes:

- Soliciting bids from suppliers (that is, querying suppliers for item price-and-availability) and awarding the order to the supplier with the best bid
- Issuing a purchase order to the selected supplier and receiving a sales order confirmation

These two business processes are connected: after GCS selects the supplier, it issues the purchase order for that supplier. These processes are triggered when anticipated or actual sales drive the need to manufacture more EnergyMiser 76 products. Increased manufacturing, in turn, drives the need to obtain more parts.

Characteristics for these business processes are not defined in the sample scenario, but analysis is recommended for factors such as:

- Frequency of ordering parts (individual parts and all parts)
- Turnaround time for selecting suppliers
- Total number of purchase orders issued
- Seasonal fluctuations in order volumes, if any

Steps for Querying Price-and-Availability

The business process for selecting a supplier involves the following steps:

1. The buyer issues a request to potential suppliers for price-and-availability on a specific item.
2. Suppliers answer the request by providing price-and-availability information to the buyer.
3. The buyer gathers the suppliers’ responses and presents them to the ordering manager.

4. The ordering manager reviews the responses and selects a supplier.

Steps for Issuing a Purchase Order

The business process for issuing a purchase order to the selected supplier involves the following steps:

1. The ordering manager issues the purchase order and notifies the selected supplier.
2. The purchase order is automatically entered into the company's enterprise resource planning (ERP) system.
3. The selected supplier returns a purchase order acknowledgment to the buyer.
4. The purchase order record is automatically updated with information from the purchase order acknowledgment (the supplier's sales order number).

Specifying Actors and Roles

In the sample WebLogic Integration application, both business processes involve trading partners acting in the buyer and supplier (seller) roles. In addition, the purchase order business process includes integration with an EIS.

Types of Actors

The GCS scenario involves human and software actors:

- Human actors:
  - the GCS purchasing agent who issues the request for item pricing and availability to suppliers
  - the supplier salesperson who responds to the request with item pricing and availability information
  - the GCS purchasing manager who selects the supplier and ends the process
- Software actor: ERP system, an Enterprise Information System (EIS) that manages the purchase order information

Actors and Roles for Selecting a Supplier

GCS plays the role of a buyer that:

- Initiates the request for pricing and availability information
- Reviews responses received from suppliers
- Selects the supplier to end the process

Each supplier plays the role of a potential seller that:

- Receives the request from the buyer
- Determines pricing and availability for the requested item
- Sends that information to the buyer

Actors and Roles for Issuing a Purchase Order

GCS plays the role of the buyer that:

- Notifies the selected supplier
- Issues the purchase order
- Enters the purchase order into the purchase order module of the ERP system
- Receives confirmation from the supplier

The GCS ERP system plays the role of information manager for the purchase order. It handles the creation of the purchase order and receipt of the acknowledgment from the supplier.

The selected supplier plays the role of seller that involves the following tasks:

- Receives the purchase order acknowledgment request from the buyer
- Provides the purchase order confirmation and supplemental information (such as the supplier's sales order number) to the buyer

Specifying Business Events

In the sample WebLogic Integration application, business events are associated with both business processes.

Events in the Supplier Selection Process

The supplier selection process includes the following events:

- Buyer requests price-and-availability from sellers.
- Sellers send price-and-availability response to the buyer.
- Buyer aggregates seller responses and selects a seller.

Events in the Purchase Order Process

The purchase order process includes the following events:

- Buyer requests a new purchase order.
- Buyer requests a purchase order acknowledgment from the selected seller.
- Seller sends acknowledgment to the buyer.

Specifying Data Flow Requirements

The sample WebLogic Integration application includes the following key data flows:

- Price-and-availability request
- Price-and-availability response
- Aggregated response
- Purchase order request
- Purchase order
- Purchase order acknowledgment

Certain information about these data flows—anticipated volume of data, peaks and lulls, seasonal patterns, and so on—is not defined in the sample WebLogic Integration application.

List of Requirements

These data flows have the following requirements:

- Business events—see “Specifying Business Events” on page 2-20.
- Conditional data
  - The list of suppliers to which the buyer sends the price-and-availability request.
  - The supplier that the buyer selects and for which the buyer issues a purchase order.
- Business rules—Rule governing the selection of a supplier, based on a comparison of the price-and-availability responses received from all supplier candidates.
- Mappings—Process of transforming data:
  - Purchase order request must be mapped from the ordering manager to the ERP system
  - Purchase order acknowledgment must be mapped from the selected seller to the ERP system
- Business transactions
  - If no responses are received from suppliers within the specified time frame, then the entire request-for-price-and-availability transaction could be rolled back.
  - If a purchase order confirmation is not received from the selected seller, then the entire purchase-order transaction could be rolled back.

Note: Transaction rollbacks for these conditions are currently not implemented in the sample WebLogic Integration application.

Data Flows in the Price-and-Availability Request Process

The price-and-availability request data flow contains the following information:

- Unique request identifier, and date and time of the request
- Item information (unique part ID, quantity requested, date and time required, and desired unit price)
- Desired location of the shipping origin
- Any supplemental notes or instructions

The price-and-availability response data flow contains the following information:

- Unique response identifier, and the date and time of the response
- Associated request identifier
- Supplier information
- Item information (unique part ID, available quantity, available date, unit price)
- Location of the shipping origin
- Any supplemental notes or instructions

The aggregated response data flow contains all the information in the price-and-availability response, but the information is collected under a common request identifier and date of request.

Data Flows in the Purchase Order Process

The purchase order request data flow contains the following information:

- Supplier name
- Quote identifier
- Item information (part number, quantity, unit price, delivery quantity, delivery date)

The purchase order data flow contains the following information:

- Purchase order header (purchase order number, issue date, status, buyer contact information, supplier information, billing information, shipping information, financial information, and total amount)
- Purchase order detail (line number, part number, part description, quantity, unit price, delivery date, and notes)

The purchase order confirmation data flow contains all the information in the purchase order, plus the sales order number and the date of the sales order.

Specifying the Quality of Service

The sample WebLogic Integration application includes Quality of Service (QoS) requirements for the following:

- Performance
- Availability and Reliability
- Response Times
- Security
- Scalability
- Logging and Nonrepudiation

Performance

While the scenario for the sample WebLogic Integration application does not include performance requirements, transaction volumes, or peak loads, we can assume that GCS needs the integration solution to provide acceptable performance at all times, regardless of the number of queries for price-and-availability sent and the number of supplier responses received.

Availability and Reliability

While there are no availability and reliability requirements for the sample WebLogic Integration application, we can safely assume that the integration solution must:

- Stay up during business hours (8am to 6pm)
- Remain available after business hours for offline processing
- Persist price-and-availability requests and purchase orders, and process them after a system restart whenever a system outage occurs
- Restart within a maximum downtime (such as two hours)

Response Times

While the sample scenario does not indicate response time requirements, we can assume that when a new purchase order is created in the ERP application, the purchase order confirmation request must be sent out almost immediately (for example, within 10 seconds).

Security

GCS has the following security requirements:

- Price-and-availability requests should be sent to authorized or approved suppliers only.
- Price-and-availability responses from suppliers are confidential. Suppliers should not know anything about competitors' bids.
- The ERP system has its own security mechanism in place to ensure data integrity and prevent unauthorized access. Any integration must have sufficient access rights to create and update a purchase order in the ERP system.

Scalability

Using WebLogic Integration in a clustered environment, GCS can scale its systems so that they handle a large number of suppliers and a substantially increased volume of bid solicitations and purchase orders.

Logging and Nonrepudiation

Purchase orders and purchase order confirmations must conform to the standards of nonrepudiation because they represent legal contracts between buyers and sellers.

Specifying the Integration Topology

Both business processes in the sample WebLogic Integration application involve communication between GCS and its trading partners across the GCS firewall. In addition, behind the GCS firewall, integration with the ERP system requires that the integration solution has network access to the ERP system.

CHAPTER

3 Designing the Integration Solution

The integration specialist builds upon the material gathered in the requirements definition phase and designs an integration solution for the WebLogic Integration environment. The integration specialist needs to know how to map the business and technical requirements to the WebLogic Integration architecture, and how to design an integration solution that makes the best use of the available WebLogic Integration architecture and features.

With one exception, the following sections describe issues to consider when designing the solution for an integration project. The final section shows how these issues relate to a high-level design specification for the sample WebLogic Integration application.

- Understanding the WebLogic Integration Architecture
- Mapping the Integration Solution Topology to WebLogic Integration
- Implementing the Business Processes
- Defining Data Transformations
- Creating a Detailed Design
- Specifying a High-Level Design for the Sample WebLogic Integration Application

Understanding the WebLogic Integration Architecture

Before attempting to design the integration solution, an integration specialist needs to understand the WebLogic Integration architecture. For an overview of the WebLogic Integration architecture, see "The Road to e-Business Integration" in *Introducing BEA WebLogic Integration*.

Specifically, an understanding of the following WebLogic Integration components is important.

Table 3-1 Key Components of the WebLogic Integration Architecture

Component	Description
Application Integration	Integrates WebLogic Integration with EIS applications using the following kinds of adapters: ■ Service adapters provide synchronous request/response integration from WebLogic Integration into an EIS application. ■ Event adapters provide unidirectional asynchronous integration from an EIS application into WebLogic Integration. Application integration adapters are configured through a Web browser. All the business events that flow through application integration adapters are defined as XML Schema Definitions (XSDs) in the WebLogic Integration repository. Application integration functionality is directly integrated with BPM through a BPM plug-in, enabling the easy definition of processes that send events to application integration service adapters and consume events generated by application integration event adapters. For more information, see "Application Integration" in *Introducing BEA WebLogic Integration*.
Business Process Management (BPM)	Supports the definition and execution of business processes. Using WebLogic Integration Studio, users can define processes graphically and monitor execution at run-time. For more information, see "Business Process Management" in *Introducing BEA WebLogic Integration*.

Table 3-1 Key Components of the WebLogic Integration Architecture

Component	Description
B2B Integration	Enables WebLogic Integration to communicate with external trading partners over the Internet. B2B integration functionality is supported for the XOCP, RosettaNet (1.1 and 2.0), and Ariba cXML protocols, all of which use XML to represent events. **Note:** The XOCP and cXML business protocols are deprecated as of this release of WebLogic Integration. For information about the features that are replacing them, see the *BEA WebLogic Integration Release Notes*. B2B is directly integrated with BPM through a BPM plug-in, enabling easy definition of processes that send and receive messages to/from external trading partners. For more information, see "B2B Integration" in *Introducing BEA WebLogic Integration*.
Data Integration	Enables the run-time translation of XML and binary data, as well as the data-mapping capability provided by Contivo Analyst, for the transformation of XML documents. WebLogic Integration supports binary-to-XML, XML-to-XML, and binary-to-binary transformations. For more information, see "Data Integration" in *Introducing BEA WebLogic Integration*.
EDI Integration	Enables WebLogic Integration to communicate with external trading partners using EDI messages. EDI Integration is combined with WebLogic Integration through application integration adapters, which enable BPM workflows to communicate using EDI. A service adapter allows the WebLogic Integration process engine to send an XML event to EDI Integration, which then maps the event to the appropriate EDI message. An event adapter allows the EDI component to receive an EDI message, map it to an XML event, and send the XML to the WebLogic Integration process engine. For more information about EDI integration, see *Using EDI with WebLogic Integration*.

Table 3-1 Key Components of the WebLogic Integration Architecture

Component	Description
WebServices	Provides sample code to support WebServices integration using WebServices technologies such as UDDI, WebServices Description Language (WSDL), and Simple Object Access Protocol (SOAP). WebLogic Integration provides the ability to invoke a WebService from a BPM workflow, enable a BPM workflow as a WebService, and to enable the Application View as a WebService. For an introduction to WebLogic WebServices, see the WebServices and XML Tech Track in the BEA dev2dev Online at the following URL: `http://dev2dev.bea.com/index.jsp`

Mapping the Integration Solution Topology to WebLogic Integration

With an understanding of the WebLogic Integration architecture, an integration specialist can map the integration topology specified in the requirements to the appropriate WebLogic Integration features. The integration specialist must determine how the following will be implemented:

- Defining the Network Topology and Scope of Administrative Control
- Defining Human User Interaction
- Integrating with EIS Applications
- Integrating with Other WebLogic Integration Clusters
- Integrating with Other Domains in the Enterprise
- Integrating with BEA TUXEDO and BEA eLink
- Integrating with External Trading Partners
- Integrating Custom Application Development

Defining the Network Topology and Scope of Administrative Control

In an enterprise with a geographically distributed WAN, separate locations might have separate domains. For example, an organization with its headquarters in New York and field offices in London and Tokyo might have three separate domains.

When designing a WebLogic Integration solution, it is essential to understand how these locations are managed because the type of domain administration determines the design approach to use.

Domains can be administered in the following ways:

- Centrally—All domains are managed centrally by an enterprise-wide IT support organization. In this environment, WebLogic Integration clusters can be installed in each domain and the WebLogic Integration solution can coordinate communication among them, as described in “Integrating with Other WebLogic Integration Clusters” on page 3-12.
- Locally—Some or all domains are administered locally by an independent IT support organization. In this environment, WebLogic Integration clusters can be installed in each domain and the WebLogic Integration solution uses B2B integration to manage communication among them, treating each site as a trading partner. This arrangement is described in “Integrating with Other Domains in the Enterprise” on page 3-13.

Defining Human User Interaction

WebLogic Integration supports two kinds of human user interaction:

- Worklist Users
- Client Users

For more information about defining integration solution requirements for human users, see “Human Users” on page 2-6.

Worklist Users

Note: The Worklist client is deprecated as of this release of WebLogic Integration. For information about the features that are replacing it, see the *BEA WebLogic Integration Release Notes*.

Worklist users are assigned work directly by the business process. Users who are assigned tasks in this way work closely with a business process. WebLogic Integration supports the assignment of tasks to users through a client application called the Worklist. Through the Worklist a task can be assigned to either a user or a *role*, an alias for a designated group of users, any of whom can perform the assigned task. The Worklist also allows you to specify that the assigned user must execute the task before the process can continue. Worklist users interact with WebLogic Integration using one of the following types of Worklist clients.

Table 3-2 Types of Worklist Clients

Worklist Type	Description
Swing GUI	Swing GUI clients use BPM functionality by using the Worklist application that is provided with WebLogic Integration.
JSP-based	JSP-based clients use BPM functionality through a Worklist client written with Java Server Pages (JSPs) that access the BPM API.
Command-line	Worklist clients use the BPM functionality by running the `CLWorklist` command.

For a JSP-based client and a command-line client, sample code is provided with the software in the `WLIHome/samples/bpm_api` directory.

For information about developing custom Worklist clients, see *Programming BPM Client Applications*.

Client Users

Client users interact with a business process indirectly, by using applications that are part of the process. For example, a sales person might log into a JSP-based application, create a sales order (which starts a workflow to process the order), and receive an e-mail confirmation.

Client users interact with the BPM through an asynchronous message-based exchange using one of three types of messages, which are described in the following sections:

- Mailbox-Based Messages
- E-mail Messages

Mailbox-Based Messages

In WebLogic Integration, the B2B integration component provides an HTML-based zeroweight client that interacts with a BPM process via database-based mailboxes. XML messages placed in a mailbox by the browser client are sent to the WebLogic Integration process engine. XML messages placed into the mailbox by the WebLogic Integration process engine can be read by the browser client. A library of JSP tags is provided to enable browser clients to interact with the mailboxes. The browser client must process the XML messages. For more information, see “Working with Zeroweight Clients” at the following URL:

`http://edocs.bea.com/wli/docs70/interm/b2bhome.htm`

Note: As of this release of WebLogic Integration, the following features described in this section are being deprecated: trading partner zeroweight clients, the B2B Mail Box, and the B2B JSP browser tag libraries. For information about the replacements for these features, see the *BEA WebLogic Integration Release Notes.*

E-mail Messages

The WebLogic Integration process engine can send e-mail messages containing either free text or XML data, as described in “Sending E-Mail Messages” in “Defining Actions” in *Using the WebLogic Integration Studio*. You can create an adapter to receive an XML message in e-mail and forward it to a BPM process (such as those represented by the start/event and event nodes in the WebLogic Integration Studio).

Integrating with EIS Applications

WebLogic Integration provides multiple mechanisms for integrating with EIS applications. The following sections describe these mechanisms:

- Application Integration Adapters
- Other Integration Techniques
- Developing a Custom Application Integration Solution

A method must be chosen for each application separately. Whenever possible, EIS applications should connect to WebLogic Integration through a LAN. For more information about defining integration requirements for EIS application integration, see "Application Integration—Enterprise Information Systems (EIS) Applications Used for Application Integration" on page 2-6.

Application Integration Adapters

The most commonly used method for integrating WebLogic Integration with external EIS applications is through an application integration adapter. Depending upon the target EIS, an off-the-shelf adapter might be available for purchase. Alternatively, a developer can write a custom adapter. The following sections describe the benefits of application integration adapters:

- Functional Capabilities
- Generic Nature
- Business-Level Interface
- Integration with BPM
- Application Development Kit

For more information about application integration adapters, see the white paper entitled *Enterprise Application Integration (EAI): Providing Stability in the Whirlwind of E-Commerce* at the following URL:

`http://www.bea.com/products/elink/EAI_business_wp.shtml`

Functional Capabilities

Because application integration adapters are based on the J2EE Connector Architecture (J2EE-CA), they provide significant functional capabilities, such as connection management, transaction management, and security management. The underlying J2EE CA engine provided by WebLogic Server supports these capabilities.

A key advantage is connection pooling. A connection to an EIS application can be an expensive resource, and the WebLogic Integration connection pooling functionality allows that resource to be shared among many WebLogic Integration requests.

Generic Nature

An application integration adapter is a generic adapter that is configured for the integration requirements. Adapters are configured through browsers. Therefore, those who configure the adapter do not need to be developers. Instead, they should be people with experience in running the application being integrated. This generic adapter design also supports dynamic integration environments.

Business-Level Interface

The interface presented to the WebLogic Integration process engine by the service and event adapters is a business-level interface (for example, *create_new_customer*), rather than an EIS application-specific API call. As a result, the BPM process designer can concentrate on the functionality of the process rather than on the mechanics of programmatically integrating with the application via the API.

All application integration service and event messages are defined as XSDs in the WebLogic Integration repository.

Integration with BPM

BPM and application integration functionality are directly integrated at both design time and run time. Application integration events and services are visible in the WebLogic Integration Studio to the workflow designer building the workflow.

Application Development Kit

WebLogic Integration provides an Application Development Kit (ADK) for developing application integration service and event adapters. For more information, see *Developing Adapters*.

Other Integration Techniques

Other integration techniques for EIS applications include:

- BPM Business Operations
- JMS Wrappers

BPM Business Operations

WebLogic Integration solutions can use BPM business operations to implement a request/response model initiated by the WebLogic Integration process engine. With business operations, a developer creates an EJB or Java class that wraps the EIS application interfaces that are used in the integration solution. Using business operations, this EJB or Java class can then be integrated with the WebLogic Integration process engine and invoked by a workflow. Integration specialists use business operations to invoke an EIS from the WebLogic Integration process engine *synchronously*.

JMS Wrappers

WebLogic Integration solutions can use JMS wrappers to implement an asynchronous integration model. A developer can create a JMS wrapper that can send messages from an EIS application to WebLogic Integration and vice versa. These messages can be created in either of the following formats:

- XML—Appropriate DTDs or XSDs should be defined in the WebLogic Integration repository.
- Binary—Data integration plug-in functionality can be used to translate between binary and XML formats.

Integration specialists use JMS wrappers to invoke an EIS from the WebLogic Integration process engine *asynchronously*.

Developing a Custom Application Integration Solution

The easiest way to provide integration with an EIS application is to purchase an off-the-shelf application integration adapter. If such an adapter is not available for custom applications, however, a custom integration solution must be developed. The integration specialist must decide whether to develop an application integration adapter, a business operation, or a JMS wrapper.

The drawback of using an off-the-shelf application integration adapter is that it might require more development effort than a custom integration. The model for which application integration adapters are designed is one in which a developer builds an adapter once and makes it available for use in many integration environments. Such a model provides the perfect rationale for developing commercial off-the-shelf adapters.

If, however, the EIS application integration requirements are simple and static (for example, if an application is designed to invoke only two API methods that are not going to be changed), then using a business operation or a JMS wrapper might involve less development effort.

Database Integration—Sample Application Integration Database Adapter

WebLogic Integration provides the source code for a sample application integration database adapter that can be used in an integration solution. The adapter supports both services and events:

- The service adapter executes configured SQL statements against the database.
- The event adapter monitors database tables for updates.

The source code to the adapter can be extended if necessary. For more information, see "The DBMS Adapter" in *Developing Adapters*.

File and FTP Integration—Sample Application Integration Binary File Adapter

The source code for a sample application integration binary file adapter is provided on the BEA dev2dev Online at the following URL:

`http://dev2dev.bea.com/index.jsp`

The sample adapter supports both services and events.

- The *service adapter* takes an XML message, translates it to binary format, using the data integration functionality provided by WebLogic Integration, and then writes the binary data to a file.
- The *event adapter* monitors a directory for a binary file, translates the binary data to XML, using the data integration functionality provided by WebLogic Integration, and then sends the XML, as a JMS message, to the WebLogic Integration process engine.

Integrating with Other WebLogic Integration Clusters

Within an enterprise, it might be necessary to integrate with other administrative domains (such as offices in other countries), as described in “Defining the Network Topology and Scope of Administrative Control” on page 3-5. If these domains are managed centrally and are visible to the integration solution over a WAN, then WebLogic Integration clusters can be installed at each site and WebLogic Integration can coordinate communication among them.

In this scenario, each WebLogic Integration cluster should be treated individually: its business processes should be customized for the requirements of the local WebLogic Integration cluster. The business processes in WebLogic Integration clusters are then integrated using XML messages that are exchanged by those business processes. The DTDs or XSDs that define these XML messages should be stored in the WebLogic Integration repository in each WebLogic Integration cluster.

For more information about WebLogic Integration clusters, see “Understanding WebLogic Integration Clusters” in *Deploying BEA WebLogic Integration*.

Integration Options

WebLogic Integration clusters can be integrated using the following WebLogic Integration features:

- B2B integration functionality using the XOCP protocol. WebLogic Integration clusters are the trading partners in an XOCP configuration, as described in *Introducing B2B Integration*.

  Note: The XOCP business protocol is deprecated as of this release of WebLogic Integration. For information about the features that are replacing it, see the *BEA WebLogic Integration Release Notes*.

- WebLogic Integration WebServices sample code, as described in the WebServices and XML Tech Track in the BEA dev2dev Online at the following URL:

  `http://dev2dev.bea.com/index.jsp`

Sample Integration

Suppose an enterprise runs a global distribution system with a central sales office in New York and regional warehouses worldwide. The New York office uses an order management system and each warehouse location runs a distribution application. Each location is running a WebLogic Integration cluster.

In this scenario, the order management application receives the order, determines which region should fulfill the order, and notifies that region. At each regional site, a workflow can handle the request and send the response back to the order management system.

Integrating with Other Domains in the Enterprise

Within an enterprise, it might be necessary to integrate with other administrative domains (such as offices in other countries) over which the integration solution has no control or into which it has no visibility. The other domains might or might not use WebLogic Integration for their own integration requirements.

WebLogic Integration domains can be integrated using the following WebLogic Integration features:

- B2B integration functionality using the XOCP protocol. WebLogic Integration clusters are the trading partners in the XOCP configuration, as described in *Introducing B2B Integration*. In these scenarios, the other domains should be regarded as trading partners that happen to be internal to the enterprise (and within the firewall). Define the business events that need to flow to and from the other domains. Because these business events are represented using XML, the associated DTDs or XSDs must be defined.

  Note: The XOCP business protocol is deprecated as of this release of WebLogic Integration. For information about the features that are replacing it, see the *BEA WebLogic Integration Release Notes.*

- WebLogic Integration WebServices sample code, as described in the WebServices and XML Tech Track in the BEA dev2dev Online at the following URL:

  `http://dev2dev.bea.com/index.jsp`

- If the other domains in the organization are using WebLogic Integration, you can integrate between WebLogic Integration clusters using the technologies described in “Integrating with Other WebLogic Integration Clusters” on page 3-12.
- If the other domains are using a different integration solution, then a custom integration must be developed before XML messages can be exchanged with that solution.

Integrating with BEA TUXEDO and BEA eLink

A WebLogic Integration solution can involve the integration of applications that are already deployed on a WebLogic Server:

- BEA WebLogic Tuxedo Connector provides integration between WebLogic Server applications and Tuxedo services. For more information, see the WebLogic Tuxedo Connector documentation at the following URL:

  `http://edocs.bea.com/wls/docs70/wtc.html`
- BEA eLink is a family of enterprise application integration (EAI) products that provides integration between WebLogic Server applications with legacy applications. For more information, see the eLink documentation at the following URL:

  `http://edocs.bea.com/elink/index.html`

Integrating with External Trading Partners

The most commonly used method for integrating with external trading partners is to develop a B2B integration solution. The following sections provide fundamental information about the B2B integration functionality provided by WebLogic Integration:

- B2B Integration Architecture
- Supported Business Protocols

B2B Integration Architecture

The architecture of a B2B integration solution depends on whether it is a value chain integration (peer-to-peer configuration) or a B2B exchange (hub-and-spoke configuration). For a detailed discussion that compares these architectures, see "Configuration Models" in "Getting Started with B2B Integration" in *Introducing B2B Integration*.

Note: Two business protocols mentioned in this section, cXML and XOCP, are deprecated as of this release of WebLogic Integration. For information about the features that are replacing them, see the *BEA WebLogic Integration Release Notes*.

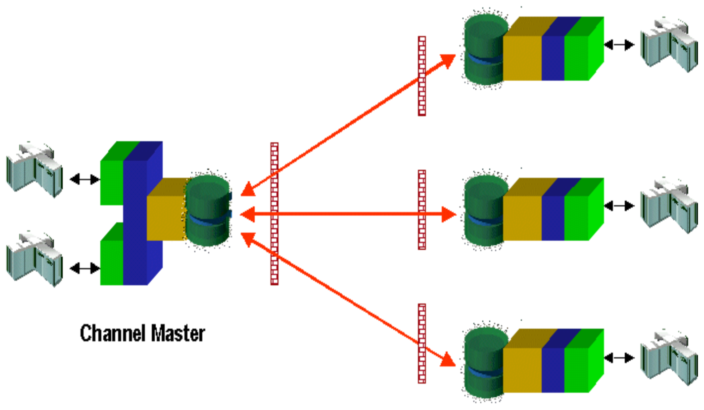
Peer-to-Peer Configuration in Value Chain Integration Solutions

A value chain integration solution involves either or both of the following types of integration:

- Supply-chain, which integrates an enterprise with its suppliers
- Demand-chain, which integrates an enterprise with its customers

The peer-to-peer configuration is used in value chain integration solutions, as shown in the following diagram.

Figure 3-1 Peer-to-Peer Configuration for Value Chain Integration Solutions

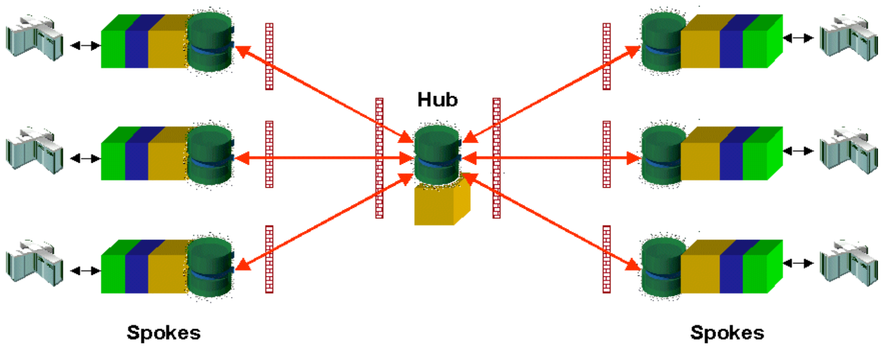


In the peer-to-peer configuration, a single trading partner (enterprise) is the Channel Master (CM)—the controlling entity to which all other trading partners (suppliers, customers, or both) are connected. This configuration model is also used when a business requires trading partners to communicate using the RosettaNet, cXML, or EDI protocol.

Hub-and-Spoke Configuration in B2B Exchange Solutions (Deprecated)

A B2B exchange solution enlists trading partners for B2B transactions. A B2B hub-and-spoke architecture is used in B2B exchange solutions, as shown in the following diagram.

Figure 3-2 Hub-and-Spoke Configuration for B2B Exchange Solutions



In the hub-and-spoke configuration, a single enterprise acts as a hub that coordinates B2B messages among enlisted trading partners (spokes). B2B applications communicate through an intermediary delivery channel using the XOCP protocol. This configuration model provides an intermediary in the message flow that performs value-added services such, as routing and filtering of messages, quality of service, and services for the trading partners in the conversation.

Hybrid Architectures

A WebLogic Integration solution might use a combination configuration that includes both peer-to-peer and hub-and-spoke configurations. This configuration model supports the exchange of messages among trading partners using XOCP for some trading partners and peer-to-peer protocols (RosettaNet, cXML, or EDI) for others.

Supported Business Protocols

A business protocol specifies the structure of business messages exchanged between trading partners, a method for processing the business messages, and a method for routing them to the appropriate recipients.

WebLogic Integration supports the following business protocols:

- ebXML—Used by a trading partner that deploys WebLogic Integration to interoperate with a trading partner that deploys the trading partner lightweight client, WebLogic Integration - Business Connect. WebLogic Integration supports the *ebXML Message Service Specification v1.0*, which defines the message enveloping and header document schema used to transfer ebXML messages with a communication protocol such as HTTP.
- XOCP (eXtensible Open Collaboration Protocol)—Used to provide a hub-and-spoke configuration and enhanced capabilities, such as message multicasting, message payload definition independence, conversation lifecycle management, and Qualities of Service (QoS) capabilities (such as message durability, timeout, retry attempts, and correlation IDs).

  Note: The XOCP business protocol is deprecated as of this release of WebLogic Integration. For information about the features that are replacing it, see the *BEA WebLogic Integration Release Notes.*
- RosettaNet—Used when trading partners require connectivity via the RosettaNet protocol, or when integration solution requirements stipulate the use of standard protocols rather than proprietary ones. For more information about RosettaNet support in B2B integration, see *Implementing RosettaNet for B2B Integration*.
- cXML (Commerce eXtensible Markup Language)—Used when Ariba connectivity is required. For more information about cXML support in B2B integration, see *Implementing cXML for B2B Integration*.

  Note: The cXML business protocol is deprecated as of this release of WebLogic Integration. For information about the features that are replacing it, see the *BEA WebLogic Integration Release Notes.*
- EDI (Electronic Data Interchange)—Used when trading partners require connectivity via EDI. For more information about EDI, see *Using EDI with WebLogic Integration*.

For a detailed discussion that compares these different business protocols, see “Supporting Business Protocols” in “Meeting the Requirements of Your E-Business” in “Overview” in *Introducing B2B Integration*.

Integrating Custom Application Development

While designing the overall architecture of a WebLogic Integration solution, the integration specialist needs to identify gaps that require the development of custom code. Application customization options include BPM plug-ins, custom adapters, and logic plug-ins.

BPM Plug-Ins

A WebLogic Integration solution might require the creation of a custom BPM plug-in that implements the BPM plug-in framework. For more information about designing and developing BPM plug-ins, see *Programming BPM Plug-Ins for WebLogic Integration*.

Custom Adapters

A WebLogic Integration solution built on application integration functionality might require the creation of a custom application integration adapter.

Custom Logic Plug-Ins (Deprecated)

A WebLogic Integration solution built on B2B integration functionality might require the creation of custom logic plug-ins, which are Java classes that perform specialized processing of business messages. Specifically, logic plug-ins insert rules and business logic at strategic locations as business messages travel through a node. For more information, see *Programming Logic Plug-Ins for B2B Integration*.

Note: Custon logic plug-ins are associated with the XOCP protocol, which is deprecated as of this release of WebLogic Integration. For information about the features that are replacing XOCP, see the *BEA WebLogic Integration Release Notes.*

Implementing the Business Processes

Once the integration solution topology has been defined, then the integration specialist can start implementing business processes. For each business process defined in the integration solution requirements, several items must be implemented, as described in the following sections:

- Business Events
- Conditional Data
- Business Rules
- Mappings
- Business Transactions

Business Events

Business events are the events that flow between the end-points of an integration solution. Some events might be sent to a process as input, while others might need to be generated within processes. In WebLogic Integration, business messages are sent in an XML or binary format.

For each business event, the integration specialist must define the following:

- A unique event name
- Metadata describing the event structure and data content
  - For XML data, the format can be either DTD or XSD.
  - For binary data, the format must be Message Format Language (MFL) produced with the data integration functionality of WebLogic Integration.

  These definitions should be stored in the WebLogic Integration repository. Whenever possible, use an event name as the name of the entity in the repository that holds the metadata. WebLogic Integration automatically stores this information in the repository when application integration adapters are being configured.

Conditional Data

Conditional data information that is required to make processing decisions, such as criteria for routing messages or for conditionally executing code. Such data is usually extracted from business messages. It might be constructed dynamically during the business process, however, from other data. Each piece of conditional data should be represented by a workflow variable.

Business Rules

Processing rules are applied to conditional data to determine the run-time execution path of the process. These rules are represented as decision nodes in a BPM workflow. In some scenarios, it is necessary to apply a number of complex conditions against a set of data. The outcome of these conditions determines the content of the workflow. A Java class should be developed to apply these conditions and return the result. This result can be used by decision nodes in the BPM workflow. The Java class you create to obtain this result can be called from the WebLogic Integration process engine as a business operation.

Mappings

Mappings define the data transformations between input and output business messages.

Business Transactions

A process might contain one or many business transactions. It must define both the business transactions and the *compensating actions*—actions that are performed when the business transaction needs to be rolled back.

Defining Data Transformations

The following sections describe data transformations for WebLogic Integration solutions:

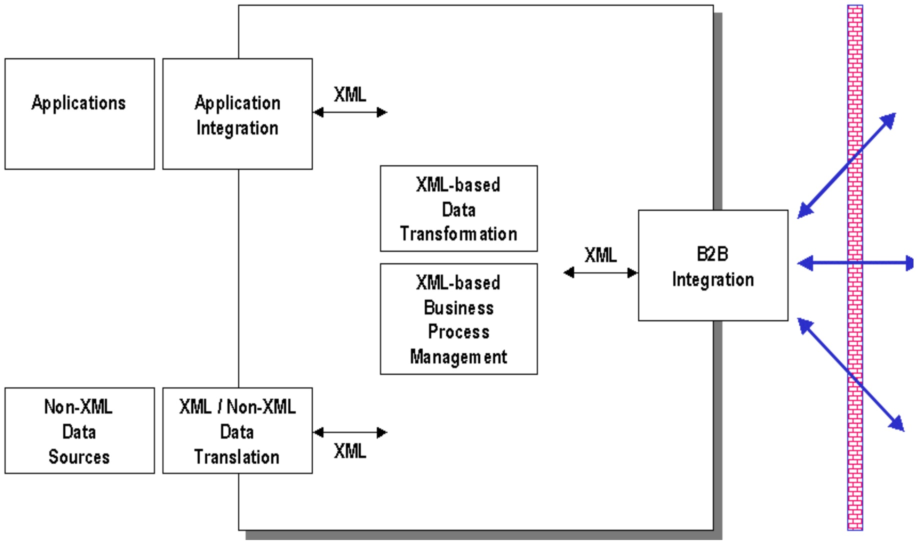
- About XML-Based Data Transformations
- Defining Binary-to-XML Transformations
- Defining Binary-to-Binary Transformations
- Defining XML-to-XML Transformations

For more information about data transformations, see "Data Integration" in *Introducing BEA WebLogic Integration* and *Translating Data with WebLogic Integration*.

About XML-Based Data Transformations

XML is the message format most commonly used in WebLogic Integration.
The following illustration shows how XML-based data transformation fits into the architecture of WebLogic Integration solutions.

Figure 3-3 XML-Based Data Transformation


Applications
Application Integration
XML
XML-based Data Transformation
XML-based Business Process Management
XML
B2B Integration
Non-XML Data Sources
XML / Non-XML Data Translation
XML

The most common sources of XML data include:

- Messages received from trading partners
- Events from backend EIS systems, such as J2EE-CA adapters
- Sources of nonXML data, such as flat files or EDI

XML is a common language with many dialects. Each source of XML data might use a different dialect, which means that a mechanism is required to handle the translation between different dialects. XSLT transformation is used as this mechanism.

For example, suppose a buyer sends a purchase order in RosettaNet format to a seller, and the seller needs to import the data into a back-end SAP ERP system, which accepts data only in SAP IDOC format. XSLT transformation can be used to convert the purchase order data from RosettaNet to IDOC.

Defining Binary-to-XML Transformations

While XML is the message format most commonly used in WebLogic Integration, at times, WebLogic Integration might need to integrate with environments in which a binary message interface is used. The data integration functionality of WebLogic Integration performs bidirectional translation between XML and binary formats. Such translations do not alter the data structure and content of the message; they change only the message format (XML or binary).

The format of the binary data is described via a Message Format Language (MFL) document created graphically using the Format Builder application. The MFL document is used by the data integration runtime class to parse and translate the binary data into an XML document.

Defining Binary-to-Binary Transformations

Binary-to-binary transformations and any combination (such as binary-to-XML and XML-to-binary) are supported by the combined used of two tools:

- WebLogic Integration for XML-to-binary translation
- XSLT for XML-to-XML transformation

Defining XML-to-XML Transformations

The *data transformation* process is one of mapping the data structure and/or content of a source message to a new data structure and/or content in a target message. WebLogic Integration supports data transformation only for XML messages, using XSLT as its data transformation mechanism.

Users can create XSL stylesheets using graphical mapping tools such as Contivo Analyst (which is bundled with WebLogic Integration), or they can create them manually. XSL stylesheets are stored in the WebLogic Integration repository.

At run time, all transformations are executed by the WebLogic Integration process engine. The input message is provided in a workflow variable. To perform the conversion, the WebLogic Integration process engine references the XSL stylesheet in the WebLogic Integration repository. It stores the output message in another workflow variable.

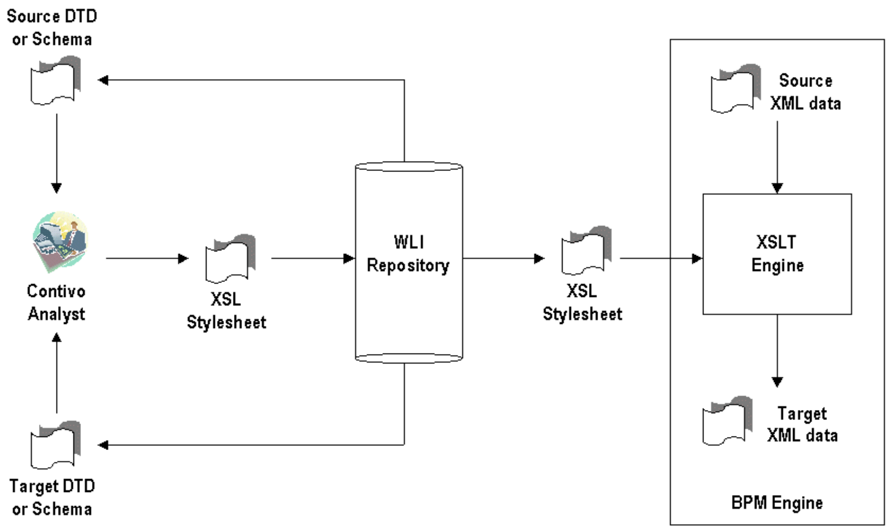
Using Contivo Analyst to Define XML-to-XML Transformations

WebLogic Integration includes Contivo Analyst, a design-time graphical mapping tool that can be used to generate XSL stylesheets. The WebLogic Integration process engine can execute these XSL stylesheets at run-time to transform a source message to a target message. The source and target messages are defined using either DTDs or XSDs, and these definitions are stored in the WebLogic Integration repository. For more information about installing, configuring, and using Contivo Analyst, see your Contivo product documentation.

Contivo Analyst Integration with the WebLogic Integration Repository

Contivo Analyst is integrated directly with the WebLogic Integration repository, as shown in the following illustration.

Figure 3-4 Contivo Integration with the WebLogic Integration Repository

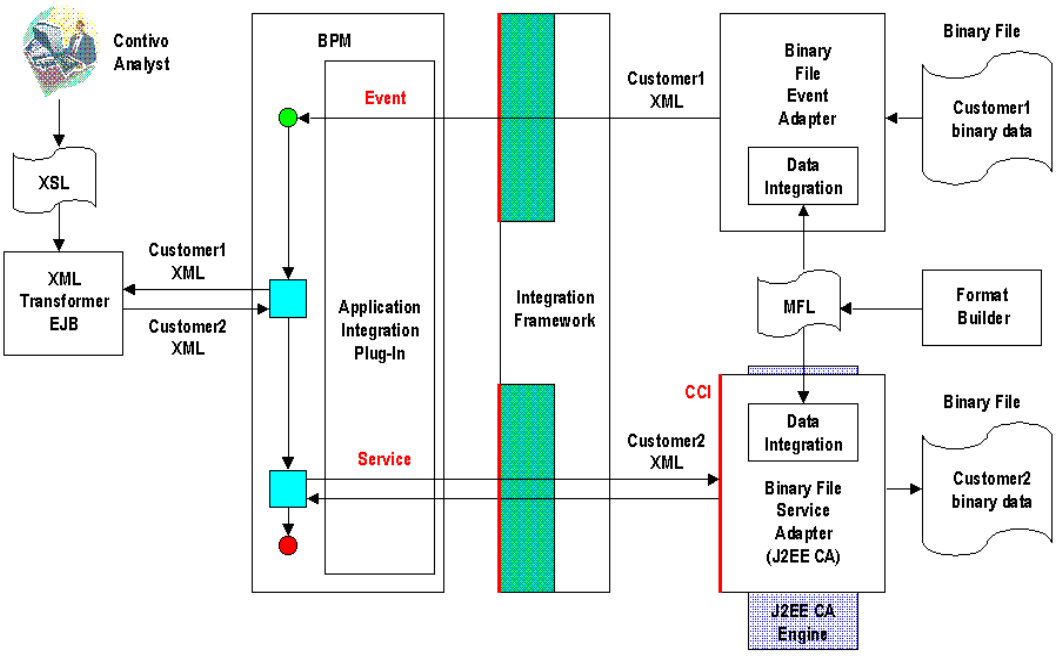


Contivo Analyst browses the repository to select source and target message definitions. Once the selected definitions are read into the Contivo Analyst, the mapping from source to target is specified graphically. When the mapping is complete, the Contivo Analyst generates the XSLT and saves it directly in the WebLogic Integration repository. The WebLogic Integration process engine executes the XSLT using the XSL Transform workflow action.

Role of Contivo Analyst in an Integration Solution

The following diagram shows how Contivo Analyst fits into a WebLogic Integration solution.

Figure 3-5 Role of Contivo Analyst in a WebLogic Integration Solution



In this illustration, Contivo Analyst generates the XSL stylesheet that the XML Transformer EJB uses to translate XML data between customers.

Note: In addition to the XSL-based translation component, this illustration shows many other elements of a WebLogic Integration solution, including BPM workflows, application integration adapters, and adapters using data integration to perform binary-to-XML transformations using a Message Format Language (MFL) document created using the Format Builder application.

Using the RosetteNet-to-IDOC transformation scenario described in “About XML-Based Data Transformations” on page 3-22, the following steps show how Contivo Analyst fits into an integration solution:

1. The developer configures the B2B engine for the business conversation with the customer.
2. The developer defines a business process that will handle the incoming purchase order (assuming that the adapter to the SAP system is configured and ready).
3. The developer loads the source (RosettaNet) and target (IDOC) formats into the Contivo Analyst.

   The formats are either DTD or XSD. They can be located in the WebLogic Integration repository or the local file system.
4. The developer builds the map in the Contivo Analyst tool by associating mapping rules from fields in the source to fields in the target.
5. When the map is complete, the developer generates the XSLT transformation file and saves it in the WebLogic Integration repository.

   The XSLT transformation file is an XML file (named with a .XSL extension) that is passed at run time to the XSLT engine, along with the input document (a purchase order).
6. In the WebLogic Integration Studio, the developer designs a business process to take the incoming purchase order, transform it from RosettaNet to IDOC, and enter it into the ERP system.

Creating a Detailed Design

After determining which WebLogic Integration components will be used in an integration solution, an integration specialist works with business analysts, developers, and system administrators to create a detailed design and begin constructing the integration solution.

The following sections provide information related to creating a detailed design for WebLogic Integration solutions:

- Using Integrated Design Tools
- Designing the Deployment Environment
- Performance Considerations in Integration Design

Using Integrated Design Tools

To help create the detailed design, use the following WebLogic Integration tools:

- WebLogic Integration Studio is used to model business processes and design and generate BPM workflows. For more information, see *Using the WebLogic Integration Studio*.
- Contivo Analyst is used to create XSL stylesheets to handle XML-to-XML transformations, as described in "Defining XML-to-XML Transformations" on page 3-23.
- Format Builder is used to create MFL documents to handle binary-to-XML transformations, as described in "Building Format Definitions" in *Translating Data with WebLogic Integration*.

Designing the Deployment Environment

An integration specialist usually designs the deployment environment before construction of the integration solution begins. An integration specialist needs this information to plan for deployment resources and costs. For information about deploying WebLogic Integration solutions, see *Deploying BEA WebLogic Integration*.

For example, to provide more workload capacity and higher scalability, an integration solution is usually deployed on a WebLogic Integration *cluster*. A cluster is a group of servers that can be managed as a single unit. For information about designing clusters, see "Designing a Clustered Deployment" in "Understanding WebLogic Integration Clusters" in *Deploying BEA WebLogic Integration*

Performance Considerations in Integration Design

The following sections describe factors that affect performance in WebLogic Integration deployments. An integration specialist needs to consider these factors when designing an integration solution:

- Message Persistence
- Message Size
- XML Parsing and Validation
- Workflows
- Application Integration
- Application Logging
- Secure Sockets Layer

For more information about tuning the run-time performance of WebLogic Integration deployments, see "Tuning Performance" in *Deploying BEA WebLogic Integration.*

Message Persistence

Message persistence is a WebLogic Integration feature that improves data recovery in the event of a hardware or network failure. If enabled, however, message persistence slows overall system performance because it adds the processing overhead required for saving or purging messages in a persistent data store (that is, in a database or file). For more information, see "Configuring Persistence and Recovery" in *Administering B2B Integration*.

Message Size

Large messages, especially when exchanged in large volumes, consume more system resources than small messages. As a result, they can slow performance. Large messages are created in two forms:

- Large XML messages
- Binary payloads

Use the WebLogic Integration B2B Console to configure large message support for B2B integration functionality. For more information, see “Defining B2B Integration Parameters” in “Configuring B2B Integration” in the *Online Help for the WebLogic Integration B2B Console*.

Note: By enabling large messages, you might slow the processing of smaller messages.

XML Parsing and Validation

XML parsing can be an expensive process and it should be avoided as much as possible. XML validation is even more expensive and it should be avoided in all deployments in which high performance is a requirement. To control XML parsing/validation, keep in mind the following considerations:

- When designing an application, avoid payload-based parsing in the messaging process wherever possible.
- WebLogic Integration provides two kinds of queues:
    - validating event queues (`com.bea.wlpi.ValidatingEventQueue`)
    - nonvalidating event queues (`com.bea.wlpi.EventQueue`).

  To use a validating event queue, pass the `-validate` parameter to the MDBGenerator utility. For more information, see “Configuring a Custom Java Message Service Queue” in “Customizing WebLogic Integration” in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

Workflows

This section provides performance tips for WebLogic Integration workflows.

Synchronous Operations

WebLogic Integration allows you to use a called subworkflow instead of a workflow called via a JMS event. In general, called workflows are faster, but they consume more resources. Higher consumption results, for example, from the fact that called workflows block the calling workflow during execution.

For deployments in which high performance is a requirement:

- If the workflow runs without waiting for events, use a called workflow.

- If the workflow waits for events, call the workflow using a JMS event.
- For the highest level of performance, perform all the processing in a single template rather than calling a subworkflow.

Asynchronous Sending of Business Messages

You can design workflows to take advantage of the high-availability features of WebLogic Integration. In fact, WebLogic Integration now requires that business messages using the RosettaNet protocol must be sent asynchronously. (Synchronously sending of business messages is no longer supported.)

Sending business messages asynchronously offers performance benefits: After a task node that sends a business message has completed execution, workflow processing can immediately pass to the next node, if desired. It is not necessary to suspend the execution of the workflow while awaiting the business message return status. This section describes how to design asynchronous message sending in your RosettaNet-based workflows.

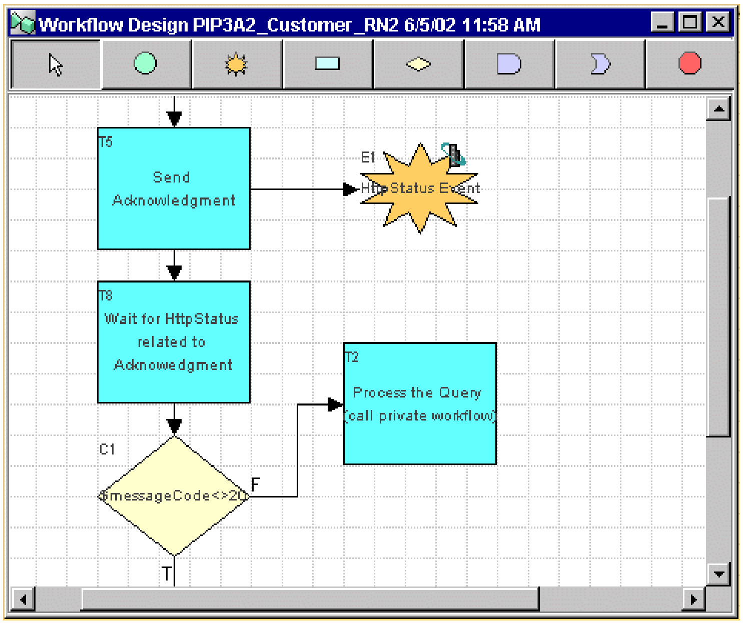
To design a workflow that sends a business message asychronously, you must define the following:

- A task node that sends the business message (as before)
- A task node that specifies a timeout value, or due date, for the HTTP status event
- An event node that receives an HTTP status event

Note: If you have RosettaNet-based workflows created with earlier versions of WebLogic Integration, you must modify the way in which they send business messages. For complete details about migrating existing RosettaNet-based workflows to the current release of WebLogic Integration, see *BEA WebLogic Integration Migration Guide*.

The following figure shows an example of how four nodes (T5, T8, C1 and E1) are associated with an asynchronous Send Business Message task in a RosettaNet-based workflow.

Figure 3-6 Send Business Message Workflow


Workflow Design PIP3A2_Customer_RN2 6/5/02 11:58 AM
T5
Send Acknowledgment
E1
HttpStatus Event
T8
Wait for HttpStatus related to Acknowedgment
T2
Process the Query (call private workflow)
C1
$messageCode<>20
F
T

For complete details about creating workflows based on the RosettaNet protocols, including instructions for specifying the asynchronous sending of business messages, see the following:

- *Implementing RosettaNet for B2B Integration*
- "Sending RosettaNet Business Messages" in "Sending and Receiving Business Messages" in *Creating Workflows for B2B Integration*.

Message Timeout

After you define and connect the nodes associated with a Send Business Message task, you typically pass execution from the task node that specifies a timeout value to a decision node. The decision node typically evaluates the HTTP status, and then determines the flow of execution as appropriate.

It is important to specify a timeout value that is large enough to avoid the “late message” problem. This problem is likely to occur if the timeout value is reached before the workflow receives the HTTP status event from the recipient trading partner. If the status event arrives after the timeout period has elapsed, the workflow ignores the status event.

As a rule of thumb, especially when dealing with remote trading partners, start with a timeout value of two hours. Testing should help you fine-tune the timeout value as appropriate.

Message Retries

Depending on the business protocol on which your workflow is based, there are different mechanisms available in WebLogic Integration that allow you to specify that a business message should be resent in the event that earlier attempt to send the business message has failed.

For workflows based on the RosettaNet protocol, the Studio allows you to design logic into your RosettaNet-based workflows that incorporates a “message resend” feature.

For ebXML-based workflows, however, you specify message retry in the B2B Console, not in the workflow. You specify this value when you configure the doc exchange for a trading partner who is participating in the ebXML conversation associated with the workflow. When you specify a message retry value, and choose OnceAndOnlyOnce in the Delivery Semantics selection box, the B2B engine uses that value to determine the number of times it should resend a business message in the event that a previous attempt has failed. Once the sending of a business message is successful, no further attempts are made. This assures that a message is sent once and only once. (If you select BestEffort in the Delivery Semantics selection box, the B2B engine ignores the retry value.)

Conditions

If performance is important, avoid using event conditions in BPM start or event nodes, if possible. For more information, see “Understanding Event Conditions” in “Defining Workflow Templates” in *Using the WebLogic Integration Studio*.

Using XPath

When using XPath, consider the following guidelines:

- When using XPath to set variables from data in an XML event, it is more efficient to set all of the variables initially in the variables table for the node that received the XML event.
- It is less efficient to set an XML variable to the payload of the XML event and then use XPath in an action that sets a workflow variable.
- When using XPath in an action that sets a workflow variable, the smaller the XML source variable, the better the performance. If XPath is used repeatedly against the subset of an XML source variable, it is better to create a temporary XML variable containing the subset and then run the XPath against the temporary variable.

Using Temporary Variables

To increase BPM performance, set temporary variables to `NULL` before a quiescent state. This prevents it from being written to disk on a transaction boundary (like an event).

Disabling Statistics Gathering

To increase BPM performance, disable statistics gathering by setting the following flag:

```
-wli.bpm.server.nostatistics=true
```

Once statistics gathering is disabled, statistics are no longer displayed in the WebLogic Integration Studio.

Application Integration

By default, application integration functionality depends on the use of transactional and persistent message delivery for asynchronous requests and responses. This type of delivery can cause unwanted messaging overhead. To improve overall system performance, disable transactional and persistent delivery using the following solutions:

- To disable transactions, open your `wlai.properties` file (in the `config/`*`domain`*`/wlai` directory) and ensure that the file includes the following line:

  ```
  wlai.jms.asyncServiceTransFlag=false
  ```

- In general, event delivery performance depends on the adapter being used to generate events. Most performance enhancements involve tuning the EIS instance and the event adapter for your EIS. Using the WebLogic Server Administration Console, change the default delivery mode of the JMS Connection Factory to *nonpersistent*.
- Asynchronous requests are processed by a per-server pool of threads. If the number of threads in this pool is lower than the number of threads that you have allocated for concurrent pending asynchronous requests, the additional requests are not serviced until a free thread can be retrieved from the thread pool.

  To solve this problem, increase the number of asynchronous processor threads by setting the `wlai.numAsyncServiceRequestProcessors` property in the `wlai.properties` file to a larger number. You might need to add this property to the `wlai.properties` file. The default value for this property is 2, which means that, by default, only 2 asynchronous service requests are processed at any given time on a given server instance.

Application Logging

To enhance overall WebLogic Integration performance in a production environment, applications should perform minimal logging. To disable logging output, see "Using Log Messages to Manage WebLogic Servers" in the *BEA WebLogic Server Administration Guide* at the following URL:

`http://edocs.bea.com/wls/docs70/adminguide/logging.html`

Secure Sockets Layer

To enhance performance in WebLogic Integration deployments in which B2B integration functionality is use, limit your use of Secure Sockets Layer (SSL) to those situations in which secure messaging is required. SSL adds processing overhead by performing such tasks as encrypting and decrypting business messages, which can slow overall system performance.

Specifying a High-Level Design for the Sample WebLogic Integration Application

The following sections discuss the high-level design of the sample WebLogic Integration application in relation to the design issues described in previous sections:

- Automating Business Processes Through the WebLogic Integration Studio
- Integrating the Supply Chain with B2B Integration
- Integrating the ERP System with Application Integration Adapters
- Managing Data Transformations with the Data Integration

These sections describe how the requirements defined in Chapter 2, "Determining Integration Solution Requirements," are mapped to components in the WebLogic Integration architecture. For a detailed description of the design for the sample WebLogic Integration application, see *Learning to Use BEA WebLogic Integration*.

Automating Business Processes Through the WebLogic Integration Studio

To automate the business processes described in "Specifying the Integration Solution and Business Processes" on page 2-17, the sample WebLogic Integration application uses the WebLogic Integration Studio to model and manage the following business processes:

- Query Price and Availability (QPA)
- Purchase Order

These business processes use private and collaborative workflows among trading partners. They encapsulate the business events described in "Specifying Business Events" on page 2-20 and the data flows described in "Specifying Data Flow Requirements" on page 2-20.

For detailed information about the workflow design, see “Business Process and Workflow Modeling” in “Understanding the Sample” in *Learning to Use BEA WebLogic Integration*.

Integrating the Supply Chain with B2B Integration

To manage business transactions with external entities (suppliers) in the supply chain, the sample WebLogic Integration application uses the B2B integration functionality provided by WebLogic Integration. Specifically, it is build on a hub-and-spoke configuration (described in “Integrating with External Trading Partners” on page 3-14) that provides mediated messaging among trading partners via the XOCP protocol. By using XOCP, the sample WebLogic Integration application meets the Quality of Service (QoS) requirements described in “Specifying the Quality of Service” on page 2-23.

Note: The XOCP business protocol is deprecated as of this release of WebLogic Integration. For information about the features that are replacing it, see the *BEA WebLogic Integration Release Notes*.

The sample WebLogic Integration application deploys four business entities: a hub, a buyer, and two supplier trading partners. However, although the scenario calls for a topology in which suppliers run their own software on separate machines outside GCS, all entities are collocated on the same machine in the sample WebLogic Integration application to simplify using the sample. For more information about running this code in a clustered environment, see *Deploying BEA WebLogic Integration*.

For detailed information about incorporating B2B integration functionality into your design, see “B2B Integration” in “Understanding the Sample” in *Learning to Use BEA WebLogic Integration*.

Integrating the ERP System with Application Integration Adapters

To integrate with the ERP purchase order system, the sample WebLogic Integration application uses application integration adapters. On the buyer side, it uses an event adapter to add the purchase order to the ERP system. It uses service adapters to update the purchase order, based on the purchase order acknowledgment received from the selected supplier.

For detailed information about the application integration adapter, see "Application and Data Integration" in "Understanding the Sample" in *Learning to Use BEA WebLogic Integration*.

Managing Data Transformations with the Data Integration

To handle XML-to-binary and binary-to-XML data transformations, the sample WebLogic Integration application uses data integration functionality. The `WLIS_SupplierPOPrivate` workflow performs the following conversions:

- For the Purchase Order: XML data-to-binary data
- For the Purchase Order acknowledgment: binary data to XML

For detailed information about data integration, see "Application and Data Integration" in "Understanding the Sample" in *Learning to Use BEA WebLogic Integration*.

Index